A Process Algebraic View of I/O Automata

by

Roberto Segala

B.S., Computer Science
University of Pisa - Italy
(1991)
Diploma in Computer Science
Scuola Normale Superiore - Pisa
(1991)

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Science in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 1992

Signature of Author____
Department of Electrical Engineering and Computer Science
September 8, 1994

Certified by____
Nancy A. Lynch
Professor of Computer Science
Thesis Supervisor

Accepted by____
Campbell L. Searle
Chairman, Departmental Committee on Graduate Students

A Process Algebraic View of I/O Automata

by

Roberto Segala

Submitted to the Department of Electrical Engineering and Computer Science

on September 8, 1994, in partial fulfillment of the

requirements for the degree of

Master of Science in Electrical Engineering and Computer Science

Abstract

The Input/Output Automata formalism of Lynch and Tuttle is a widely used framework for the specification and verification of concurrent algorithms. Unfortunately, it has never been provided with an algebraic characterization, a formalization which has been fundamental for the success of theories like CSP, CCS and ACP. We present a many-sorted algebra for I/O Automata that takes into account notions such as interface, input enabling, and local control. It is sufficiently expressive for representing all finitely branching transition systems, hence all I/O automata with a finitely branching transition relation. Our presentation includes a complete axiomatization of the quiescent preorder relation over recursion free processes with input and output. Finally, we give some example specifications and use them to show the methodology of verification based on our algebraic approach.

Thesis Supervisor: Nancy A. Lynch

Title: Professor of Computer Science

Keywords: I/O Automata, Process Algebras, Distributed Systems, Software Specification

Contents

Acknowledgements

This work would not have been written without the suggestions of Frits Vaandrager. His pioneering paper "On the relationship between Process Algebras and Input/Output Automata" and his sharp comments had a strong influence on the achieved results. Particularly important was the discussion that lead us to the choice of using many-sorted algebras. Vaandrager also suggested me to use the handshaking protocol as an example application.

I am also grateful to Rocco De Nicola for advising me on a previous work done in Pisa that constituted the base for this thesis. In particular he continuously encouraged me to search for a clean way of representing I/O automata within process algebras, even after I moved to Boston. The algebra I present in this thesis, in fact, has passed through several adjustments.

Extremely useful was the interacting environment available at MIT. The interactions with other students and professors has given me the possibility of a continuous feedback of the results I was gradually achieving. In particular I want to thank Prof. Albert Meyer for his helpful criticism on draft versions of this work. I actually spent many hours in his office.

I also want to thank my advisor Nancy Lynch for giving me the possibility of proceeding along the lines of my preceding work. She read my preliminary work providing me with many questions that influenced my research progress. One of the most important suggestions I received from Nancy is to give a double view of my axioms: axioms can be seen as theorems about I/O automata or as properties of syntactical entities.

Once again the frequent interaction between Nancy Lynch, Albert Meyer and me was fundamental for achieving my goals, i.e., trying to reduce the distance between two different, but similar, schools of concurrency by speaking a language comprehensible to everyone.

Chapter 1

Introduction

The Input/Output Automata [LT87, Sta84, Jon85, Jon87] is a widely used and deeply investigated formalism for specifying and verifying concurrent systems. Unfortunately, it has never been provided with an algebraic characterization, a mathematical formalization that has been fundamental for the success of theories like CSP, CCS and ACP [Hoa85, Mil89, Hen88, BW90]. The goal of this thesis is to improve our understanding of the intricacies of I/O automata by describing them as a process algebraic theory. This will permit algebraic manipulation and provide an alternative to the commonly used verification method based on possibilities mapping.

We start by designing an algebra that incorporates the fundamental features of I/O automata of Lynch and Tuttle [LT87] and captures the essential role of concurrent composition, hiding and renaming of I/O automata. Our design aims at maintaining minimality of operators and universal expressivity with respect to the I/O automata we can represent. We base our characterization on the following basic features of I/O automata:

1. *explicit interfacing*: a transition-invariant interface is associated with each process;
2. *input/output distinction*: a clear distinction is made between output actions that are locally controlled and input actions that are globally controlled;
3. *input enabling*: input actions are enabled in every state;
4. *local control*: each action is under the control of at most one process.

Clearly this list is not exhaustive, and for the sake of simplicity we choose at this stage to avoid considering important issues such as fairness.

The operators in our calculus associate distinct sets of input and output actions (interfaces) with each process. This captures a critical aspect of I/O automata, namely the distinction between input and output actions. To associate an interface to a process we use many-sorted algebras: each sort stands for an interface. This permits dealing with partial operators in a clean way. As an example consider the parallel composition operator. To comply with the requirement that each action is under the control of at most one process, two processes that have common output actions cannot be composed in parallel. Many-sorted algebras permit capturing this restriction by defining the parallel operator as a family of sorted operators, one for each pair of compatible interfaces.

Our research continues a line of investigation initiated by Vaandrager in [Vaa91]. That investigation was deliberately done in a simple setting where no explicit interface is associated to processes, and in which input enabling is obtained by means of self loops. No axiomatization was proposed in [Vaa91]. Indeed, the behavioral relation we use for comparing systems is the *quiescent preorder* of [Vaa91] (definition 2.2.4 of chapter 2). The main idea of the quiescent preorder is that a quiescent trace leads system to a state from which only input actions are enabled. Moreover the preorder is given by external and quiescent trace inclusion. The quiescent preorder is a restriction to finite traces of the fair preorder of [LT87], and we see it as a stepping stone toward the study of fairness sensitive semantics.

An important property we require of our calculus is substitutivity of the quiescent preorder. One of our guides for achieving substitutivity is again [Vaa91] where, in the style of [De 84, De 85b, GV89, BIM90], restrictions to the inference rules of a generic Structured Operational Semantics [Plo81] are investigated to guarantee substitutivity of the quiescent and fair preorders. Our calculus, however, does not completely fit Vaandrager's format and thus new congruence proofs are needed.

A key issue in defining our I/0 calculus is the way input enabling is enforced. We present our choice with the support of an example. Consider process $P = a.e$, which is able to perform an action a and then behave like e. If the system is input enabled, the above process must be able to perform any other input action different from a. We considered two different possible choices,

1. *Angelic*: Unexpected inputs are ignored and give rise to self-loops. For example, system $P = a.e$, after accepting any input b different from a, behaves as before, and is ready to accept the a-action.

2. *Demonic*: Unexpected inputs are considered as catastrophic; after any unexpected input a system moves to a special state Ω from which any behavior is possible. Thus, $P = a.e$, after any b-action different from a, moves to Ω.

The Angelic choice was made by Vaandrager in [Vaa91]; here, we support the Demonic one. In our view, the prefixing operator specifies the behavior of P only for action a and says nothing about input actions different from it. By interpreting this in the field of I/O automata we have that an implementation of P should be correct independently of the behavior it exhibits when provided with any input action different from a. Since the relation we use to compare processes is the quiescent preorder, moving to a special state Ω from which any behavior is possible makes the above interpretation possible. Due to this basic choice, our calculus will be called the *Demonic calculus of I/O Automata* (DIOA).

This demonic approach has been partially influenced by the *Receptive Process Theory* (RPT) of Mark Josephs [Jos92]. However, the semantics of RPT provided by Mark Josephs is denotational, and like CSP, is described by means of sets of failures, traces and divergencies. The handling of underspecification is even more demonic than ours; underspecification is propagated backward, i.e., if a process P can perform an output action o and move to the equivalent of an Ω state, then the whole P is equivalent to Ω.

For DIOA, we propose a set of sound algebraic laws that are *complete* with respect to the quiescent preorder for recursion-free processes. The completeness result is achieved through reduction to a special normal form in which the parallel operator is used in a restricted way. Particularly important for our result is an operator representing internal choice. It does not fit Vaandrager's general format and forces us to prove substitutivity of our preorder explicitly.

We give a dual view of the algebraic laws: from one point of view a law is a theorem about I/O automata; from the other point of view a law is a statement about the relationship between two syntactic entities. The dual view of the laws has the advantage of separating the properties of the model chosen for DIOA (I/O automata) from the properties based on the syntactic structure of the expressions. The main difference between the two points of view lies in the way that side conditions are defined, i.e., in the way in which the conditions for the validity of a law are expressed: according to the first point of view a side condition is defined in terms of the semantics associated with an expression; according to the second point of view a side condition is defined in terms of the syntactic structure of an expression.

Finally, we present two simple example specifications and implementations within DIOA in which the quiescent preorder is used as an implementation relation and we outline a methodology for verification based on our algebraic laws. The examples suggest an alternative to the commonly used verification method based on possibilities mapping and show that, in some cases, algebraic reasoning might be simpler than directly searching for a mapping between states of processes.

The rest of the thesis is organized as follows: Chapter 2 contains some preliminary definitions; Chapter 3 presents the Demonic Calculus of I/O Automata; Chapter 4 presents a set of algebraic theorems for DIOA, corresponding to the first point of view of the algebraic laws; Chapter 5 provides an axiomatization of the quiescent preorder over DIOA expressions that is complete for recursion-free processes; Chapter 6 presents some example specifications; Chapter 7 presents some concluding remarks and some suggestions for further work. The end of the thesis contains an appendix with the formal definition of DIOA and the complete list of the axioms that are introduced in chapters 4 and 5.

Chapter 2

Preliminaries

In this chapter we give a general introduction to the formalisms we are comparing. Section 2.1 formally introduces I/O automata giving their definition together with some of the main features and some of the commonly used preorder relations. Section 2.2 introduces process algebras and other new preorder relations. The preorder relations of Section 2.2 are the process algebraic version of the relations presented in Section 2.1.

2.1 I/O automata

In this section we formally introduce I/O automata whose complete formal definition is given in [LT87]. One of the basic concepts is the notion of action signature. Basically an action signature represents the interface of an automaton with the external environment.

Definition 2.1.1 (action signature) Given three disjoint sets in, out and int we refer to the triple (in, out, int) as an *action signature* S. The sets in, out and int are respectively denoted by $in(S)$, $out(S)$ and $int(S)$. The entire set of actions $in \cup out \cup int$ is denoted by $acts(S)$. The set of *external actions* $in \cup out$ is denoted by $ext(S)$. Finally the set of *locally controlled actions* $int \cup out$ is denoted by $local(S)$. ∎

We can now formally define an I/O automaton.

Definition 2.1.2 (input-output automaton) An *input-output automaton* A is a tuple $A = (Q, Q_0, S, t, P)$ where

- Q is a set of states and is referred to as $states(A)$,
- $Q_0 \subseteq Q$ is the set of start states and is referred to as $start(A)$,
- S is an action signature and is referred to as $sig(A)$,
- $t \subseteq Q \times acts(S) \times Q$ with the property that $\forall q \in Q, a \in in(S)\, \exists q' \in Q : (q, a, q') \in t$. It is referred to as $steps(A)$, and
- P is a partition of $local(S)$ and is referred to as $part(A)$.

A step $(q, a, q') \in steps(A)$ is conventionally denoted by $q \xrightarrow{a} q'$. ■

The difference between classical automata and I/O automata is essentially in the differentiation of the actions given by the action signature, the constraint that the transition relation is always defined for input actions, and the presence of the partition P of the locally controlled actions. We will discuss the role of P when introducing the notion of fair execution. For the moment we concentrate on executions.

Definition 2.1.3 (executions and schedules) Given an automaton A, an *execution fragment* is a finite sequence $q_0 a_0 q_1 \cdots a_k q_k$ or infinite sequence $q_0 a_0 q_1 a_1 q_2 \cdots$ of alternating states and actions such that $(q_i, a_i, q_{i+1}) \in steps(A)$ for every i. An *execution* is an execution fragment beginning with a start state (i.e., $q_0 \in start(A)$). The *schedule* of an execution x is the subsequence of actions appearing in x. It is denoted by $sched(x)$. The executions and schedules of an automaton A are denoted respectively by $execs(A)$ and $scheds(A)$. ■

Usually it is necessary to deal with subsets of an automaton's executions or schedules. For this reason we define the notion of execution module and schedule module. The basic idea is that an execution module simply represents a set of executions while a schedule module represents a set of schedules.

Definition 2.1.4 (execution and schedule modules) An *execution* module E is a triple $E = (Q, S, e)$ where Q is a set of states, S is an action signature and e is a set of executions with actions in $acts(S)$ and states in Q. They are referred to as $states(E)$, $sig(E)$ and $execs(E)$.

A *schedule module* C is a pair $C = (S, c)$ where S is an action signature and c is a set of schedules with actions in $acts(S)$. They are referred to as $sig(C)$ and $scheds(C)$. ■

Given an automaton A there is a natural execution module $Execs(A)$ associated with it.

$$Execs(A) = (states(A), sig(A), execs(A)).$$

Given an execution module E there is a natural schedule module $Scheds(E)$ associated with it.

$$Scheds(E) = (sig(E), scheds(E)).$$

I/O automata, execution modules and schedule modules are collectively referred to as objects and denoted by O.

As a last step, we restrict the observation of an automaton to its external actions.

Definition 2.1.5 (external schedule module) An *external action signature* is an action signature consisting only of external actions. An *external schedule module* is a schedule module with an external action signature.

The external action signature of a signature S is $(in(S), out(S), \emptyset)$, i.e., S without internal actions; given a sequence y of actions and a set of actions X we denote by $y\lceil X$ the subsequence of y consisting only of actions of X.

The external schedule module of an object O, denoted by $External(O)$, is the external schedule module with the external action signature of O and the schedules $\{y\lceil ext(O) : y \in Scheds(O)\}$. ■

We can now define the first notion of equivalence for I/O automata.

Definition 2.1.6 (unfair equivalence) The *unfair behavior* of an object O, which is denoted by $Ubeh(O)$, is the external schedule module $External(O)$. Two objects O and P are said to be *unfairly equivalent*, $O \equiv_U P$, iff $Ubeh(O) = Ubeh(P)$. ■

This relation is an equivalence relation and turns out to be a congruence for the operators defined over objects. There are three operations defined over objects: hiding, renaming and parallel composition.

Definition 2.1.7 (hiding) Given an object O and a set of actions $I : I \cap in(O) = \emptyset$, we define the object $Hide_I(O)$ to be the object differing from O in that

- $out(Hide_I(O)) = out(O) \backslash I$, and
- $int(Hide_I(O)) = int(O) \cup (acts(O) \cap I)$.

■

The effect of the hiding operator is to hide some locally controlled actions to the external environment. The only difference from the argument of the operator and its resulting object is that the signature is changed. Executions and schedules are exactly the same. Clearly external schedules change. The definition of the hiding operator of [LT87] does not contain the restriction that $I \cap in(O) = \emptyset$, but it is immediate to observe that the operator is not closed for I/O automata if we allow to hide input actions: $part(A)$ would not be a partition of $local(A)$ any more.

Definition 2.1.8 (renaming) An injective mapping f is applicable to an object O if $acts(O) \subseteq dom(f)$. Given an automaton A and a mapping f applicable to A we define $f(A)$ to be (Q, Q_0, S, t, P) where

- $Q = states(A)$, $Q_0 = start(A)$,
- $in(S) = f(in(A))$, $out(S) = f(out(A))$, $int(S) = f(int(A))$,
- $t = \{(q, f(a), q') : (q, a, q') \in steps(A)\}$, and
- $P = \{(f(a), f(a')) : (a, a') \in part(A)\}$.

■

The definition above can be easily reformulated for execution modules and schedule modules. The effect of the renaming operator is simply to rename actions.

Definition 2.1.9 (composition of I/O automata) A set of action signatures $\{S_i : i \in I\}$ is called *compatible* iff for all $i, j \in I$ we have

1. $out(S_i) \cap out(S_j) = \emptyset$, and
2. $int(S_i) \cap acts(S_j) = \emptyset$.

In general the objects $\{O_i : i \in O\}$ are compatible iff their action signatures are compatible. The composition $S = \prod_{i\in I} S_i$ of compatible action signatures $\{S_i : i \in I\}$ is defined to be the action signature with

1. $in(S) = \bigcup_{i\in I} in(S_i) - \bigcup_{i\in I} out(S_i)$,
2. $out(S) = \bigcup_{i\in I} out(S_i)$, and
3. $int(S) = \bigcup_{i\in I} int(S_i)$.

The composition $A = \prod_{i\in I} A_i$ of compatible automata $\{A_i : i \in I\}$ is defined to be the automaton with

1. $states(A) = \prod_{i\in I} states(A_i)$,
2. $start(A) = \prod_{i\in I} start(A_i)$,
3. $sig(A) = \prod_{i\in I} sig(a_i)$,
4. $part(A) = \bigcup_{i\in I} part(A_i)$,
5. $steps(A) = \{((q_i)_{i\in I}, a, (q_i')_{i\in I}) : \forall i \in I$
   (a) $a \in acts(A_i) \Longrightarrow (q_i, a, q_i') \in steps(A_i)$
   (b) $a \notin acts(A_i) \Longrightarrow q_i = q_i' \qquad \}$.

■

Composition of automata is of fundamental importance because it exactly characterizes the way I/O automata communicate. The compatibility conditions state that internal actions can not interact and that every action can be controlled by at most one process. The transition function states that all processes must synchronize on common actions. The following two definitions extend the composition operator to execution modules and schedule modules.

Definition 2.1.10 (composition of execution modules) The composition $E = \prod_{i\in I} E_i$ of compatible execution modules $\{E_i : i \in I\}$ is defined as follows:

- $states(E) = \prod_{i \in I} states(E_i)$,
- $sig(E) = \prod_{i \in I} sig(E_i)$.

Given a state $s = (s_i)_{i \in I}$ of the composition, we define $s \lceil E_i = s_i$. Given a sequence $x = q_0 a_0 q_1 \cdots$ of states and actions of E, we define $x \lceil E_i$ to be the sequence obtained from x by removing all $a_j q_j$ if $a_j \notin acts(E_i)$ and replacing the remaining s_j by $s_j \lceil E_i$.

- $execs(E) = \{x = q_0 a_0 q_1 \cdots : \forall i \in I \, x \lceil E_i \in execs(E_i) \wedge (a_j \notin acts(E_i) \implies s_j \lceil E_i = s_{j+1} \lceil E_i)\}$.

■

Definition 2.1.11 (composition of schedule modules) The composition $C = \prod_{i \in I} C_i$ of schedule modules $\{C_i : i \in I\}$ is defined as follows:

- $sig(C) = \prod_{i \in I} sig(C_i)$,
- $scheds(C) = \{y : \forall i \in I \, y \lceil C_i \in scheds(S_i)\}$.

■

The following facts hold for I/O automata and show that the definitions above are well given. The interested reader may refer to [LT87] for the proofs.

Proposition 2.1.12 *Let $\{A_i : i \in I\}, A$ be compatible automata, $\{E_i : i \in I\}, E$ be compatible execution modules, $\{C_i : i \in I\}, C$ be compatible schedule modules and $\{O_i : i \in I\}$ be objects. Then*

1. $Execs(\prod_{i \in I} A_i) = \prod_{i \in I} Execs(A_i)$,
2. $Scheds(\prod_{i \in I} E_i) = \prod_{i \in I} Scheds(E_i)$,
3. $External(\prod_{i \in I} C_i) = \prod_{i \in I} External(C_i)$,
4. $Ubeh(\prod_{i \in I} O_i) = \prod_{i \in I} Ubeh(O_i)$,

5. $Execs(Hide_J(A)) = Hide_J(Execs(A))$,

6. $Scheds(Hide_J(E)) = Hide_J(Scheds(E))$,

7. $External(Hide_J(C)) = External(Hide_J(External(C)))$,

8. $Execs(f(a)) = f(Execs(A))$,

9. $Scheds(f(e)) = f(Scheds(e))$,

10. $External(f(C)) = f(External(C))$.

■

A side effect of input enabling consists of the possible prevention of a system from performing locally controlled actions by means of an infinite sequence of input actions. This case is avoided by restricting the observations to fair executions. In the following definition we use the partitions of the locally controlled actions for the first time.

Definition 2.1.13 (fair executions) A *fair execution* of an automaton A is an execution x such that for all $X \in part(A)$

- If x is finite then no action of X is enabled from the final state of x
- If x is infinite then either actions from X appear infinitely often in x or states from which no action of X is enabled appear infinitely often in x

A finite fair execution is also said to be *quiescent*. ■

The notion of fairness defined above recalls weak fairness [Fra86], but the two concepts are different. In [Fra86] fairness is considered for each action, while in I/O automata fairness is considered for locally controlled actions only. Moreover, instead of considering single actions, fairness is defined in terms of sets of actions within I/O automata. The idea behind the partition of locally controlled actions is that every element of the partition represents the actions under the control of a component of the global system. In this way the notion of fair turn is expressed, i.e., each component that is continuously willing to perform a locally controlled action will eventually do so. The following two propositions are proven in [LT87].

Proposition 2.1.14 *If x is a finite execution of an automaton A, then x can be extended to a fair execution $x a_1 q_1 \cdots$ of A in which every a_i is a locally controlled action of A.* ∎

Proposition 2.1.15 *For all compatible automata $\{A_i : i \in I\}$, $Fair(\prod_{i \in I} A_i) = \prod_{i \in I} Fair(A_i)$ where $Fair(A_i)$ is the execution module having $fair(A_i)$ as its set of executions and $fair(A_i)$ is the set of fair executions of A_i.* ∎

We can now define the fair behaviors of an automaton as $Fbeh(A) = External(Fair(A))$ and give a new equivalence relation that turns out to be a weak congruence for the automata's operators, i.e., a relation that is substitutive for the I/O automata operators whenever these operators are defined for all the considered expressions.

Definition 2.1.16 (fair equivalence) Two objects O, P are *fair equivalent* ($O \equiv_F P$) iff $Fbeh(O) = Fbeh(P)$. ∎

With the concept of fair trace it is possible to introduce the notion of *implementation*. An object O_1 implements an object O_2 if they both have the same action signature and $Fbeh(O_1) \subseteq Fbeh(O_2)$. Trivial implementations are avoided by input enabling and fairness. These two concepts, in fact, state that a process must accept all stimuli from the external environment and must perform its output actions whenever it has the possibility to do so, i.e., it must give an answer when requested.

On the base of the previous discussion we can define three main relations between I/O automata that will be used throughout the rest of the thesis.

Definition 2.1.17 (preorder relations) Given an object O, let $Quiescent(O)$ be the set of quiescent executions of O and let $Qbeh(O) = External(Quiescent(O))$. Finally, let $FinUbeh(O)$ be the set of finite unfair behaviors of O.
The *external trace preorder* on objects is defined as follows: $O \sqsubseteq_{ET} P$ iff

1. O and P have the same external action signature and

2. $FinUbeh(O) \subseteq FinUbeh(P)$.

The *quiescent preorder* on objects is defined as follows: $O \sqsubseteq_Q P$ iff

1. $O \sqsubseteq_{ET} P$ and

2. $Qbeh(O) \subseteq Qbeh(P)$.

The *fair preorder* on objects is defined as follows: $O \sqsubseteq_F P$ iff

1. O and P have the same external action signature and

2. $Fbeh(O) \subseteq Fbeh(P)$.

The kernels of $\sqsubseteq_{ET}, \sqsubseteq_Q$ and $\sqsubseteq_F$ are respectively called *external trace equivalence*, *quiescent equivalence* and *fair equivalence*.

- $O \equiv_{ET} P$ iff $O \sqsubseteq_{ET} P$ and $P \sqsubseteq_{ET} O$,

- $O \equiv_Q P$ iff $O \sqsubseteq_Q P$ and $P \sqsubseteq_Q O$,

- $O \equiv_F P$ iff $O \sqsubseteq_F P$ and $P \sqsubseteq_F O$.

■

A method to prove that an object O_1 implements another object O_2 makes use of the notion of a possibilities mapping. The main idea of a possibilities mapping is to map every reachable state s of O_1 onto a set of states $h(s)$ of O_2 in such a way that every step $s_1 \xrightarrow{a} s_2$ of O_1 can be performed from any state of $h(s_1)$. The steps of O_2 must end in a state of $h(s_2)$. For a formal definition of possibilities mapping and its use the reader is referred to [LT87].

2.2 Process Algebras

The main idea of Process Algebras is the existence of some basic processes and some fundamental operators modeling operations such as sequential composition, parallel composition, nondeterministic composition and synchronization. A process is represented by an expression which is built inductively from the basic processes and the fundamental operators. The semantics of each expression is given in terms of an underlying model which may vary from algebra to algebra.

Particularly important is the way in which processes are identified in the underlying model. The equivalence (preorder) relations defined on the underlying models induce equivalence (preorder) relations on the interpreted expressions. The next step is then to define a sound and possibly complete proof system over the expressions with the result that the relationship between expressions can be proven by means of pure algebraic analysis.

One of the first process algebras was the calculus of *Communicating Sequential Processes* (CSP) [Hoa85]. CSP has a large amount of operators and its semantics is given in terms of *traces* (sequences of actions a process can perform) and *refusal sets* (sets of actions that a process may refuse to perform). An action represents a visible move of a system.

Another algebra is the *Calculus of Communicating Systems* (CCS) [Mil89]. The underlying model of CCS is given by labeled transition systems (LTS), which are state machines with a labeled transition relation. A LTS is associated with a CCS expression by means of an operational semantics as described in [Plo81]. The standard notion of equivalence for CCS is bisimulation [Par81].

In this thesis we concentrate on the LTS approach by using I/O automata as underlying model and we analyze a particular preorder relation which is connected to the fair preorder of I/O automata. For a better understanding of other different existing relations the interested reader is referred to [De 87] and [Gla90].

We now introduce the main notions for the definition of a process algebra based on the LTS approach. We start with the notion of signature.

Definition 2.2.1 (signatures and terms) Let $\mathcal{S}$ be a set of *sorts* ranged over by $s, s_1, s_2, \ldots$ A *signature element* is a triple $(f, s_1 s_2 \cdots s_n, s)$ consisting of a function symbol f, a sequence of sorts $s_1 \cdots s_n : s_i \in \mathcal{S}, i = 1, \ldots, n$, and a single sort $s \in \mathcal{S}$. s is called the sort of the signature element and n is its arity. In a signature element (c, λ, s), c is often referred to as a constant symbol of sort s. A *signature* is a pair $\Sigma = (\mathcal{S}, \mathcal{O})$ consisting of a set of sorts $\mathcal{S}$ and a set of signature elements $\mathcal{O}$. We denote sort and function symbols of a signature Σ by $sorts(\Sigma)$ and $op(\Sigma)$. The set of *terms* over Σ, is denoted by $T(\Sigma)$. The set of terms of a particular sort $s \in \mathcal{S}$ are denoted by $T(\Sigma)_s$. ■

A signature represents the basic processes (constants) and the operators that are considered as fundamental ($(f, s_1 s_2 \cdots s_n, s)$ is an operator taking n processes respectively of sort $s_1 \cdots s_n$

as arguments and giving back a process of sort s. Well known calculi like CCS are one-sorted. We presented the more general many-sorted definition because we use sorts to model interfaces associated with processes.

The following definition introduces the notions of substitutive relation.

Definition 2.2.2 (substitutivity) Let Σ be a signature and let $\mathcal{R}$ be a relation over $T(\Sigma) \times T(\Sigma)$. $\mathcal{R}$ is *substitutive* iff for each signature element $(f, s_1 s_2 \cdots s_n, s)$ of Σ and each t_i, t'_i of sort s_i,

$$t_1 \; \mathcal{R} \; t'_1, \ldots, t_n \; \mathcal{R} \; t'_n \Longrightarrow f(t_1, \ldots, t_n) \; \mathcal{R} \; f(t'_1, \ldots t'_n).$$

■

We proceed by formally defining a calculus.

Definition 2.2.3 (calculi) Let A be a given set of labels and let Σ be a signature. A transition rule has the form

$$\frac{t_1 \xrightarrow{a_1} t'_1, \ldots, t_n \xrightarrow{a_n} t'_n}{t \xrightarrow{a} t'}$$

where $t_i, t'_i \in T(\Sigma)$, $t, t' \in T(\Sigma)$, $a_i \in A$ and $a \in A$. The elements $t_i \xrightarrow{a_i} t'_i$ are called the *premises* and $t \xrightarrow{a} t'$ is called the conclusion. The interpretation of a rule is that, whenever the transitions of the premises are possible, the transition of the conclusion is possible. Transition rules can be parameterized using variables in their terms. A *calculus*, is a triple $P = (\Sigma, A, R)$ where Σ is a signature, A is a set of labels and R is a set of transition rules. ■

We extend the transitions to sequences of labels in the obvious way by saying that $t \xrightarrow{a_1 \cdots a_n} t'$ iff $\exists t_1, \ldots, t_{n-1} : t \xrightarrow{a_1} t_1 \longrightarrow \cdots \longrightarrow t_{n-1} \xrightarrow{a_n} t'$.

We finally adapt two of the preorder relations of section 2.1 to the process algebraic framework. Fairness is not considered at this stage. The definition of the quiescent preorder is an adaptation to the many-sorted framework of the definition of [Vaa91]. In particular we identify sorts with action signatures; i.e., we assume the existence of a bijective mapping from sorts to action signatures. We use the same relation symbols we used in section 2.1 to emphasize the fact that we are expressing the same notions in different formalisms. We also abuse notation by writing $ext(e)$ when we mean $ext(S)$ where S is the action signature associated with the sort of e.

Definition 2.2.4 (preorder relations) Given a many-sorted calculus with input and output actions, the set of enabled actions from an expression e is defined as

$$\{a | \exists e' : e \xrightarrow{a} e'\}.$$

An expression e is *quiescent* if it only enables input actions.

The set of (finite) external traces of an expression e of sort S is defined as

$$etraces(e) = \{h \lceil ext(S) | \exists e' : e \xrightarrow{h} e'\}$$

where h denotes a sequence of actions and $h \lceil A$ is the projection of h on A.

The set of quiescent traces of an expression e of sort S is defined as

$$qtraces(e) = \{h \lceil ext(S) | \exists e' : e \xrightarrow{h} e', quiescent(e')\}.$$

The *external trace preorder* $\sqsubseteq_{ET}$ is defined as follows: $e_1 \sqsubseteq_{ET} e_2$ iff

1. e_1 and e_2 have the same external action signature and
2. $etraces(e_1) \subseteq etraces(e_2)$.

The *quiescent preorder* $\sqsubseteq_Q$ is defined as follows: $e_1 \sqsubseteq_Q e_2$ iff

1. $e_1 \sqsubseteq_{ET} e_2$ and
2. $qtraces(e_1) \subseteq qtraces(e_2)$.

The kernels of $\sqsubseteq_{ET}$ and $\sqsubseteq_Q$ are respectively called *external trace equivalence* and *quiescent equivalence*.

- $e_1 \equiv_{ET} e_2$ iff $e_1 \sqsubseteq_{ET} e_2$ and $e_2 \sqsubseteq_{ET} e_1$,
- $e_1 \equiv_Q e_2$ iff $e_1 \sqsubseteq_Q e_2$ and $e_2 \sqsubseteq_Q e_1$.

■

Chapter 3

A Calculus of Demonic I/O Automata

This chapter introduces a calculus for I/O automata following the demonic approach. The calculus is many sorted and each sort represents an action signature consisting of input and output actions and a single internal action τ. In the I/O automaton model action signatures may have more than one internal action, and the reason for that is to have flexibility in expressing fairness with respect to different internal tasks. Since we do not address the issue of fairness in this thesis, we present only the simple calculus with a single internal action. At the end of this chapter we give an idea of how to extend the calculus to handle multiple internal actions.

The rest of the chapter is organized as follows: Section 3.1 presents the definition of DIOA and discusses its operators; Section 3.2 presents I/O automata definitions of the operators of DIOA; Section 3.3 presents a construction associating an I/O automaton with each DIOA expression; Section 3.4 presents an I/O automata interpretation of recursion, a tool that is used for the definition of DIOA; Section 3.5 discusses the problem of introducing multiple internal actions.

3.1 The definition of DIOA

In this section we present the *calculus of Demonic I/O automata* (DIOA); it permits representing any finitely branching I/O automaton [LT87]. Moreover, the operational semantics of the

Name	Op.	Domain	Range	Restrictions
quiescent	nil_S	λ	S	
omega	Ω_S	λ	S	
prefixing	$a._S$	S	S	$a \in ext(S)$
ichoice	$\oplus_S$	S, S	S	
echoice	${}_I+_J^S$	S, S	S	$I, J \subseteq in(S)$
parallel	${}_{S_1}\|_{S_2}$	S_1, S_2	S_3	$out(S_1) \cap out(S_2) = \emptyset$ $out(S_3) = out(S_1) \cup out(S_2)$ $in(S_3) = (in(S_1) \cup in(S_2)) \backslash out(S_3)$
hiding	τ_I^S	S	S'	$I \subseteq out(S), S' = (in(S), out(S) \backslash I)$
renaming	ρ_S	S	S'	for each injective $\rho : acts(S) \longrightarrow acts(S')$ $S' = (\rho(in(S)), \rho(out(S)))$
process	X_S	λ	S	$X_S \in \mathcal{X}_S$

Table 3.1: The signature of DIOA

operators of DIOA specifies the same transition trees as of the corresponding operators for I/O automata.

Table 3.1 presents the signature for DIOA. The sort symbols associated with the operators range over all possible action signatures with a single internal action τ if no additional restrictions are mentioned. Thus, rather than a single operator (e.g. parallel, renaming, etc.) we actually have a family of operators parameterized on the sorts of the operands. To avoid heavy notation we will drop the sort indexes from the operators whenever the sorts are evident. Indeed all non-constant operators are uniquely determined by the sorts of their operands. As additional simplification we will represent action signatures as pairs (in, out) since the set of internal actions is fixed to be $\{\tau\}$. In choosing the operators we had in mind two major goals: representing the three main operators of I/O automata (i.e., parallel, hiding and renaming) and expressing a sufficient number of transition trees. The second goal is achieved through prefixing, external choice and recursion; the internal choice operator will turn out to be useful for proving completeness of axioms. Recursion is obtained in a De Simone style [De 84, De 85b].

We assume the existence of a countable set $\mathcal{X}_S$ of process variables for each sort S and the existence of a declaration mapping E associating a guarded expression of sort S to each process variable of $\mathcal{X}_S$. An expression e is guarded if each process variable occurs in the scope of a prefixing operator.

Table 3.2 presents the transition rules for DIOA; some comments follow:

- quiescent expression "nil_S":

  This expression models a quiescent automaton, where no output actions are enabled. It has a transition to Ω_S for each input action of sort S. Each input action of S, in fact, is unspecified in nil_S. No output is permitted.

- omega expression "Ω_S":

  This expression models the unspecified process, for which everything is possible. It has a self-loop for each action of S with the consequence that any trace with actions from S is an external trace of Ω_S. An additional transition to nil_S (rule $\mathbf{ome}_2$) makes any trace a quiescent trace of Ω_S. Note that the use of rule $\mathbf{ome}_2$ is the only way to move Ω to a quiescent state.

- prefixing operator "$a.$":

  In our interpretation $a\,.\,e$ specifies the behavior of a process only when it first performs action a. For all other input actions there is a transition to Ω, meaning that every choice of implementation is correct.

- internal choice operator "$\oplus$":

  The expression $e \oplus f$ can move either to e or f with an internal action (rules $\mathbf{ich}_{1,2}$ resembling the $\oplus$ of [DH87]) or behave like e or f (rules $\mathbf{ich}_{3,4}$ resembling the CCS $+$). Rules $\mathbf{ich}_{3,4}$ are necessary for input enabledness. This is an additional difference from IOC of Vaandrager [Vaa91] since the internal choice operator of IOC has self loops for any input action. The choice of using rules $\mathbf{ich}_{3,4}$ implies that the external and quiescent traces of $e_1 \oplus e_2$ are obtained by unioning those of e_1 and those of e_2. Note that none of the four rules can be eliminated; elimination of $\mathbf{ich}_{3,4}$ would cause loss of input enabling,

nil	$nil_S \xrightarrow{a} \Omega_S$	$\forall a \in in(S)$			
ome$_1$	$\Omega_S \xrightarrow{a} \Omega_S$	$a \in ext(S)$	**ome**$_2$	$\Omega_S \xrightarrow{\tau} nil_S$	
pre$_1$	$a \,._S\, e \xrightarrow{a} e$		**pre**$_2$	$a \,._S\, e \xrightarrow{b} \Omega_S$	$\forall b \in in(S) \backslash \{a\}$
ich$_1$	$e_1 \oplus_S e_2 \xrightarrow{\tau} e_1$		**ich**$_2$	$e_1 \oplus_S e_2 \xrightarrow{\tau} e_2$	
ich$_3$	$\dfrac{e_1 \xrightarrow{a} e_1'}{e_1 \oplus_S e_2 \xrightarrow{a} e_1'}$	$\forall a \in in(S)$	**ich**$_4$	$\dfrac{e_2 \xrightarrow{a} e_2'}{e_1 \oplus_S e_2 \xrightarrow{a} e_2'}$	$\forall a \in in(S)$
ech$_1$	$\dfrac{e_1 \xrightarrow{a} e_1'}{e_1 \;{}_I{+}_J^S\; e_2 \xrightarrow{a} e_1'}$	$\forall a \in I \cup out(S)$			
ech$_2$	$\dfrac{e_2 \xrightarrow{a} e_2'}{e_1 \;{}_I{+}_J^S\; e_2 \xrightarrow{a} e_2'}$	$\forall a \in J \cup out(S)$			
ech$_3$	$e_1 \;{}_I{+}_J^S\; e_2 \xrightarrow{a} \Omega_S$	$\forall a \in in(S) \backslash (I \cup J)$			
ech$_4$	$\dfrac{e_1 \xrightarrow{\tau} e_1'}{e_1 \;{}_I{+}_J^S\; e_2 \xrightarrow{\tau} e_1' \;{}_I{+}_J^S\; e_2}$				
ech$_5$	$\dfrac{e_2 \xrightarrow{\tau} e_2'}{e_1 \;{}_I{+}_J^S\; e_2 \xrightarrow{\tau} e_1' \;{}_I{+}_J^S\; e_2'}$				
tau$_1$	$\dfrac{e \xrightarrow{a} e'}{\tau_I^S(e) \xrightarrow{a} \tau_I^S(e')}$	$a \notin I$	**tau**$_2$	$\dfrac{e \xrightarrow{a} e'}{\tau_I^S(e) \xrightarrow{\tau} \tau_I^S(e')}$	$a \in I$
rho	$\dfrac{e \xrightarrow{a} e'}{\rho_S(e) \xrightarrow{\rho(a)} \rho_S(e')}$				
par$_1$	$\dfrac{e_1 \xrightarrow{a} e_1' \qquad e_2 \xrightarrow{a} e_2'}{e_1 \;{}_{S_1}\|_{S_2}\; e_2 \xrightarrow{a} e_1' \;{}_{S_1}\|_{S_2}\; e_2'}$				
par$_2$	$\dfrac{e_1 \xrightarrow{a} e_1'}{e_1 \;{}_{S_1}\|_{S_2}\; e_2 \xrightarrow{a} e_1' \;{}_{S_1}\|_{S_2}\; e_2}$	$a \in acts(S_1) \backslash ext(S_2)$			
par$_3$	$\dfrac{e_2 \xrightarrow{a} e_2'}{e_1 \;{}_{S_1}\|_{S_2}\; e_2 \xrightarrow{a} e_1 \;{}_{S_1}\|_{S_2}\; e_2'}$	$a \in acts(S_2) \backslash ext(S_1)$			
rec	$\dfrac{e \xrightarrow{a} e'}{X \xrightarrow{a} e'}$	if $X \stackrel{\text{def}}{=} e$			

Table 3.2: The transition rules for DIOA

while elimination of $\mathbf{ich}_{1,2}$ could give rise to problems whenever λ is a quiescent trace of one argument but not of the other one.

- choice operator "${}_I{+}_J$":

  The arguments of ${}_I{+}_J$ can perform an input action a only if a is in the corresponding parameter I or J (rules $\mathbf{ech}_{1,2}$). For input actions not in $I \cup J$ there is a transition to Ω (rule $\mathbf{ech}_3$). The choice context is not resolved with internal actions (rules $\mathbf{ech}_{4,5}$). This is essentially Vaandrager's choice operator. It would have been nice to define a CCS-like external choice operator without parameters, however our attempts have failed in the sense that we have not been able to achieve substitutivity for the quiescent preorder without using I and J. See Remark 3.1.8 for an example.

- hiding, renaming and parallel operators "$\tau_I, \rho, \|$":

  They are in direct correspondence with the operators of I/O automata. In particular, the constraints on the sorts for the parallel operator guarantee that actions are under the control of at most one process. The transition rules for the parallel operator state that all processes synchronize on common actions and evolve independently on the others. Note that, although processes synchronize on common actions, the communication is asynchronous since at most one process has the control of each action. The restrictions on hiding and renaming are directly inherited from I/O automata. Injectivity of ρ is required to guarantee distributivity and the restriction on hiding is kept to avoid unnecessary complications.

Below, a few basic properties of DIOA are listed.

Definition 3.1.1 (sort consistency) A many-sorted calculus is *sort consistent* if the sort of every expression is invariant under transition. ■

Proposition 3.1.2 *DIOA is sort consistent.* ■

Definition 3.1.3 (input enabledness) An expression e is *input enabled* if $\forall e' | \exists_{h \in acts(e)^*} e \xrightarrow{h} e', in(e) \subset enabled(e')$. A many-sorted calculus with interfaces associated with expressions is *input enabled* if each expression is input enabled. ■

Proposition 3.1.4 *DIOA is input enabled.* ■

Theorem 3.1.5 (substitutivity) *External trace preorder and quiescent preorder are substitutive for DIOA.* ■

The proofs of the above results are standard and can be done by cases analysis. For the substitutivity theorem we cannot use Vaandrager's results [Vaa91] since the internal choice operator does not fit Vaandrager's general format.

Remark 3.1.6 It is possible to characterize each DIOA expression in terms of the external and quiescent traces it exhibits. The inductive definition is as follows:

- $etraces(\Omega_S) = qtraces(\Omega_S) = ext(S)^*$,
- $etraces(nil_S) = qtraces(nil_S) = \{\lambda\} \cup \{at | a \in in(S), t \in ext(S)^*\}$,
- $etraces(a\,.\,e) = \{\lambda\} \cup \{at | t \in etraces(e)\} \cup \{bt | b \in in(S)\backslash\{a\}, t \in ext(e)^*\}$,
  $$qtraces(a\,.\,e) = \begin{cases} \{\lambda\} \cup \{at | t \in qtraces(e)\} \cup \{bt | b \in in(S)\backslash\{a\}, t \in ext(e)^*\} & \text{if } a \in in(e), \\ \{at | t \in qtraces(e)\} \cup \{bt | b \in in(S)\backslash\{a\}, t \in ext(e)^*\} & \text{if } a \notin in(e), \end{cases}$$
- $etraces(e \oplus f) = etraces(e) \cup etraces(f)$,
  $qtraces(e \oplus f) = qtraces(e) \cup qtraces(f)$,
- $$\begin{aligned} etraces(e\ {}_I{+}_J\ f) = \{\lambda\} & \cup \{at | a \in I \cup out(e), at \in etraces(e)\} \\ & \cup \{at | a \in J \cup out(f), at \in etraces(f)\} \\ & \cup \{at | a \in in(S)\backslash(I \cup J), t \in ext(e)^*\}, \end{aligned}$$
  $$qtraces(e\ {}_I{+}_J\ f) = \begin{cases} (\{\lambda\} \cap qtraces(e) \cap qtraces(f)) \cup \\ \quad \{at | a \in I \cup out(e), at \in qtraces(e)\} \cup \\ \quad \{at | a \in J \cup out(f), at \in qtraces(f)\} \cup \\ \quad \{at | a \in in(S)\backslash(I \cup J), t \in ext(e)^*\} \end{cases}$$
- $etraces(\tau_I(e)) = \{t\lceil(ext(e)\backslash I) | t \in etraces(e)\}$,
  $qtraces(\tau_I(e)) = \{t\lceil(ext(e)\backslash I) | t \in qtraces(e)\}$,
- $etraces(\rho(e)) = \{\rho(t) | t \in etraces(e)\}$,
  $qtraces(\rho(e)) = \{\rho(t) | t \in qtraces(e)\}$,

- $etraces(e\|f) = \{t \in ext(e\|f)^* | t\lceil ext(e) \in etraces(e), t\lceil ext(f) \in etraces(f)\}$,
  $qtraces(e\|f) = \{t \in ext(e\|f)^* | t\lceil ext(e) \in qtraces(e), t\lceil ext(f) \in qtraces(f)\}$.

Remark 3.1.7 The main difference between internal and external choice can be seen by means of an external observer. Consider processes

$$P_1 \stackrel{\text{def}}{=} a \,.\, b \,.\, nil \;{}_{\{a\}}{+}_{\emptyset}\; nil \quad \text{and}$$

$$P_2 \stackrel{\text{def}}{=} a \,.\, b \,.\, nil \oplus nil$$

where a is an input action and b is an output action. Consider an external observer O performing an output action a for then waiting for an input action b. If O is interacting with P_1 it will always receive the b-signal after performing the a-action since the choice context of P_1 is resolved when O provides a; if O is interacting with P_2 then the system could send any signal to O since P_2, while receiving a, can either move according to $a \,.\, b \,.\, nil$ or nil. In other words P_2 has decided internally how accepting action a.

The reader might think that $e \oplus f$ is equivalent to $e \;{}_A{+}_A\; f$ where $A = in(e)$. This fact, unfortunately, is false since there are possibilities of discrepancies when considering the quiescence of λ. The difference can be noted by letting O interact respectively with $a \,.\, (b \,.\, nil \;{}_{\emptyset}{+}_{\emptyset}\; nil)$ and $a \,.\, (b \,.\, nil \oplus nil)$. In the first case O will always receive the $b - signal$ while, in the second case, the interacting process may internally decide not to perform the b-move.

Remark 3.1.8 There are some immediate questions about the definition we have given for the choice operators:

(a) why did we choose only to allow internal and external choice of expressions with the same action signature?

(b) why did we choose to use two parameters I, J for the external choice operator?

The answer to question (a) is strictly related to sort consistency. Suppose we allowed the sum (external choice) of expressions with different signatures and consider

$$P_1 \equiv a.nil_{(\emptyset,\{a\})} \;{}_{\emptyset}{+}_{\emptyset}\; b.nil_{(\emptyset,\{b\})}$$

$$P_2 \equiv a.nil_{(\emptyset,\{a,b\})}\ {}_{\emptyset}{+}_{\emptyset}\ b.nil_{(\emptyset,\{a,b\})}$$

where every pair associated with nil represents its action signature (recall that the pair (in, out) represents an action signature having input actions in, output actions out and internal action τ). It is reasonable to say that the output actions of P_1 are $\{a, b\}$, hence $traces(P_1) = \{\lambda, a, b\} = traces(P_2)$. Consider now

$$P_3 \equiv nil_{(\{a\},\emptyset)}$$

It is immediate to see that $in(P_1 \parallel P_3) = in(P_2 \parallel P_3) = \emptyset$ and that $traces(P_2 \parallel P_3) = \{\lambda, a, b\}$. On the other hand P_1 "loses" the output action a after performing action b because there is no reason to consider a an output action of $nil_{(\emptyset,\{b\})}$. In particular a becomes an input action if P_1 is composed with P_3, hence $traces(P_1 \parallel P_3) = \{\lambda, a, b, ba, baa, bab, \ldots\}$ and trace preorder is not substitutive. By means of some changes on the external signature it might be possible to define a calculus with dynamic signatures (i.e., a calculus that is not sort consistent) that is substitutive for trace preorders, but this topic goes beyond the scope of this thesis.

For point (b) one might like to define an unparameterized choice operator and implicitly treat transitions to Ω. Consider for example the expression $a \,.\, e_1 + b \,.\, e_2$ where a, b are input actions and consider another input action c of e_1. When provided with a the system should evolve to e_1 since the behavior for a is specified by $a \,.\, e_1$; when provided with b the system should evolve to e_2 since the behavior for b is specified by $b \,.\, e_2$; when provided with c the system should move to Ω since the behavior for c is not specified neither by $a \,.\, e_1$ nor by $b \,.\, e_2$. It is easy to see that external and quiescent trace preorders are not substitutive for $+$. Consider for example the signature $S = (\{a\}, \{b\})$. We can easily check that

$$nil \equiv_Q a \,.\, \Omega$$

since nil moves to Ω with action a, but

$$a \,.\, nil + nil \not\equiv_Q a \,.\, nil + a \,.\, \Omega$$

since ab is a quiescent trace of the right process but not of the left one. Process nil, in fact, does not specify the behavior for action a, hence $a \,.\, nil + nil$, when provided with a, should move to nil from which action b is not enables; on the other hand the behavior for a is specified by $a\,.\,\Omega$, hence $a\,.\,nil + a\,.\,\Omega$ can move to Ω with action a and then perform action b before moving to nil. Unfortunately we have not been able to find an unparameterized choice operator for which the quiescent preorder is substitutive.

3.2 DIOA operators for I/O automata

In the previous section we have defined the transition rules for the renaming, hiding and parallel operators of DIOA in such a way that they behave in the same way as the correspondent operators of I/O automata with a single internal action. We also have defined another set of operators (prefixing, internal choice, external choice) and a set of basic expressions (nil and Ω) in order to have a sufficient expressive power.

In this section we define a new set of operators for I/O automata with one internal action in such a way that they have the same behavior as of the prefixing, internal choice and external choice operators of DIOA. We analyze each single operator: let $A = (Q_A, Q_A^0, S_A, t_A, P_A)$ and $B = (Q_B, Q_B^0, S_B, t_B, P_B)$.

- prefixing operator "$a.$":

  The automaton $a\,.\,A$, where $a \in acts(S_A)$, is defined to be

$$(Q_A \cup \{q\} \cup Q_\Omega, \{q\}, S_A, t', P_A)$$

  where Q_Ω is the set of states of the unspecified automaton and

$$\begin{aligned} t' &= t \\ &\cup \{(q, a, q_A) | q_A \in q_A^0\} \\ &\cup \{(q, b, q_\Omega^0) : b \in in(S_A) \backslash \{a\}\} \\ &\cup t_\Omega \end{aligned}$$

  where q_Ω^0 is the initial state of the unspecified automaton and t_Ω is the transition relation for the unspecified automaton. The unspecified automaton is formally defined in the next section. Here we just assume that it can be defined.

- internal choice operator "$\oplus$"

  The automaton $A \oplus B$, where $S_A = S_B$, is defined to be

$$(Q_A \cup Q_B \cup \{q\}, \{q\}, S_A, t_A \cup t_B \cup t', P_A)$$

  where

$$\begin{array}{rcl} t' & = & \{(q, \tau, q_A) | q_A \in Q_A^0\} \\ & \cup & \{(q, \tau, q_B) | q_B \in Q_B^0\} \\ & \cup & \{(q, a, q'_A) | a \in in(S_A) \text{ and } \exists q_A \in Q_A^0 : (q_A, a, q'_A) \in t_A\} \\ & \cup & \{(q, a, q'_B) | a \in in(S_B) \text{ and } \exists q_B \in Q_B^0 : (q_B, a, q'_B) \in t_B\} \end{array}$$

- external choice operator "${}_I{+}_J$"

  The automaton $A \;{}_I{+}_J\; B$, where $S_A = S_B$ and $I, J \subseteq in(S_A)$, is defined to be

$$(Q_A \cup Q_B \cup Q_A \times Q_B \cup Q_\Omega, Q_A^0 \times Q_B^0, S_A, t', P_A)$$

  where

$$\begin{array}{rcl} t' & = & t_A \\ & \cup & t_B \\ & \cup & t_\Omega \\ & \cup & \{(q_A \times q_B, a, q'_A) | (q_A, a, q'_A) \in t_A, a \in I \cup out(S_A), q_B \in Q_B\} \\ & \cup & \{(q_A \times q_B, a, q'_B) | (q_B, a, q'_B) \in t_B, a \in J \cup out(S_B), q_A \in Q_A\} \\ & \cup & \{(q_A \times q_B, a, q_\Omega^0) | a \in in(S_A) \backslash (I \cup J), q_A \in Q_A, q_B \in Q_B\} \\ & \cup & \{(q_A \times q_B, \tau, q'_A \times q_B) | (q_A, \tau, q'_A) \in t_A, q_B \in Q_B\} \\ & \cup & \{(q_A \times q_B, \tau, q_A \times q'_B) | (q_B, \tau, q'_B) \in t_A, q_A \in Q_A\} \end{array}$$

  Note that the above definition might contain many unreachable states.

The substitutivity result of Theorem 3.1.5 and the compositionality results of Remark 3.1.6 are trivially valid also for the new operators defined over I/O automata.

We can also define a transition relation directly over I/O automata as follows:

$$(Q_A, Q_A^0, S_A, t_A, P_A) \xrightarrow{a} (Q_A, \{q\}, S_A, t_A, P_A)$$

iff $\exists q_A \in Q_A^0 : (q_A, a, q) \in t_A$. Finally we can define the notion of quiescent automaton as follows: $(Q_A, Q_A^0, S_A, t_A, P_A)$ is quiescent iff $\exists q \in Q_A^0 | q$ is quiescent. The main result, relating the DIOA operators with the I/O automata operators, is then the following:

Proposition 3.2.1 (transition rules for I/O automata) *For every I/O automata operator op of arity n, the transition relation of the composition of n automata $A_1, \ldots, A_n$ is completely determined in terms of the transition relations of $A_1, \ldots, A_n$ by using the transition rules for DIOA. More precisely, if $\exists A | op(A_1, \ldots, A_n) \xrightarrow{a} A$ according to the transition relation defined on I/O automata, then $\exists A' \equiv_Q A | op(A_1, \ldots, A_n) \xrightarrow{a} A'$ according to the transition rules of DIOA and vice versa.*

Proof. Simple cases analysis for each operator. ∎

The above proposition says that we can use the transition rules for DIOA in order to determine the behavior of the composition of simpler automata. Moreover it confirms the fact that the definitions of the operators for I/O automata are consistent with the definitions of the corresponding operators of DIOA.

3.3 DIOA expressions and I/O automata

In this section we define what it means for an expression to represent an I/O automaton by explicitly constructing the automaton associated with it.

Definition 3.3.1 Given an expression e of sort s, the automaton $Aut(e)$ associated with e is defined to be $Aut(e) = (S, Q, q_0, t, P)$ where S is the action signature associated with sort s, Q is the set of reachable states from e, q_0 is e, t is the transition relation associated with e, and $P = \{local(S)\}$. ∎

The fact that $Aut(e)$ is an I/O automaton is a direct consequence of the input enabling and sort consistency properties of DIOA expressions. The definition of the partition P of the locally controlled actions of S is arbitrary since we do not deal with fairness.

We now state two important propositions showing the consistency of the definitions we have given in this chapter.

Proposition 3.3.2 *Given a DIOA expression e,*

1. $etraces(e) = Ubeh(Aut(e))$ *and*

2. $qtraces(e) = Qbeh(Aut(e))$.

Proof. Direct consequence of Definition 3.3.1. ■

Proposition 3.3.3 *Aut is a morphism from DIOA expressions to I/O automata.*

Proof. We prove the proposition for the internal choice operator. The proof for the other operators is similar.

$$\begin{array}{lcl} Ubeh(Aut(e \oplus f)) & = & \text{by Proposition 3.3.2} \\ etraces(e \oplus f) & = & \text{by Remark 3.1.6} \\ etraces(e) \cup etraces(f) & = & \text{by Proposition 3.3.2} \\ Ubeh(Aut(e)) \cup Ubeh(Aut(f)) & = & \text{by Remark 3.1.6 applied to I/O automata} \\ Ubeh(Aut(e) \oplus Aut(f)). & & \end{array}$$

The case for the quiescent behaviors is similar. ■

Proposition 3.3.3 says that DIOA operators are preserved by the mapping Aut. For example

$$Aut(e \oplus f) \equiv_Q Aut(e) \oplus Aut(f)$$

where the left $\oplus$ is the internal choice operator of DIOA and the right $\oplus$ is the internal choice operator of I/O automata.

3.4 Recursion and I/O automata

How can recursion be interpreted within I/O automata? A definition of the form $X \stackrel{\text{def}}{=} E(X)$ can be interpreted as an equation between I/O automata meaning that the automaton X and the automaton $E(X)$ have to be quiescent trace equivalent. In other words the automaton

X has to be a fixpoint of the equation $X \equiv_Q E(X)$. It could be the case, however, that the equation has more than one fixpoint, therefore we need a method for choosing a particular fixpoint of an equation.

A natural fixpoint that can be considered is $Aut(X)$ where X and $E(X)$ are viewed as DIOA expressions. In Chapter 4 we provide a theorem about the uniqueness of the fixpoint for a set of equations.

3.5 Dealing with multiple internal actions

DIOA does not completely capture the features of the I/O automaton model since it is defined on signatures with one only internal action. The choice of this restricted set of action signatures is due to the fact that we do not address the problem of fairness within this thesis.

It is not difficult to expand DIOA in such a way that it deals with multiple internal actions. Two main consequences must be kept into consideration: the preorder relations will be defined between expressions with different sorts (all sorts with the same external action signature) and substitutivity will be no longer valid (if $P \equiv Q$ it might happen that there is a process C such that $P \| C$ is legal while $Q \| C$ is not legal). The new property that is valid is weak substitutivity, i.e., two equivalent processes cannot be distinguished in any context in which they can both be inserted.

The problem of defining calculi with multiple internal actions is completely addressed in [Seg91] where Vaandrager's work [Vaa91] is extended to the many-sorted setting. In [Seg91] there is also the extended version of an angelic calculus of I/O automata (called IOA).

Chapter 4

Algebraic theorems for the Quiescent Preorder

This chapter presents a set of theorems about I/O automata and the operators defined in Chapter 3. A theorem is a statement about the relationship between two automata where each automaton is represented by expressions with free variables. Each variable is meant to represent an I/O automaton. An example of a theorem is

$$e \equiv_Q e \oplus e \tag{4.1}$$

stating that an automaton e is equivalent to the internal choice composition of e with itself. In other words $\oplus$ is idempotent.

Not all theorems, however, can be just expressed as a relationship between two expressions. For example, it is not true in general that the automaton e is equivalent to the automaton $e \,{}_I{+}_J\, e$. The above equivalence is valid only if a particular property $P(e)$ is valid for the set of external and quiescent traces of e. The statement of the theorem is then

$$e \equiv_Q e \,{}_I{+}_J\, e \text{ if } P(e) \tag{4.2}$$

meaning "if the automaton e satisfies the property P then $e \equiv_Q e \,{}_I{+}_J\, e$". The condition expressed by the property P is called *side condition.*

From the algebraic point of view, however, the above theorems have to be interpreted as assertions about DIOA expressions meaning, for example, that the DIOA expression e is equivalent to the DIOA expression $e \oplus e$. In the case of DIOA a theorem is called *axiom*, and an axiom is said to be sound for the I/O automaton model if it is stating a true property of the automata associated with the related expressions.

An additional property of axioms is that they have to be model independent, i.e., they have to be stated purely in terms of the syntactic structure of an expression without using any semantical reasoning. In particular theorem (4.2) cannot be directly interpreted as an axiom since its side condition is not expressed in terms of the syntactic structure of e, rather in terms of the semantics associated with e.

To view theorem (4.2) as an axiom we need a syntactic characterization p of P or a sound proof system for P. In this thesis we pursue the approach of the syntactic characterization p of P. It might not be the case that a syntactic property p equivalent to P can be defined, therefore in general we introduce a property p such that $p(e)$ implies $P(e)$ and we write a real axiom

$$e \equiv_Q e \;{}_I{+}_J\; e \text{ if } p(e). \tag{4.3}$$

In this thesis we want to keep a clear distinction between theorems and axioms. Theorems are helpful for people working with I/O automata only since they provide a set of manipulation rules for I/O automata; axioms, on the other side, are useful for algebraists since they permit to capture the essence of the quiescent preorder just by means of syntactical analysis.

In accordance to the dual view theorems/axioms, this chapter deals with theorems only by providing their statements based on semantic side conditions. The next chapter, instead, provides the axiomatic view of the theorems of this chapter by providing syntactic approximations of the side conditions used in this chapter.

The rest of this chapter is organized as follows: Section 4.1 presents some auxiliary semantic functions which are used for the formulation of the side conditions for the theorems; Section 4.2 presents general theorems concerning I/O automata where the auxiliary functions are those of Section 4.1. The theorems of Section 4.2 will be converted into axioms in the next chapter; Section 4.3 presents some tools for dealing with recursively defined automata. Since the soundness proofs of the theorems are standard, we just provide the actual soundness proofs of some of them.

4.1 Auxiliary functions

In this section we introduce and justify some auxiliary functions that are useful for the formulation of the theorems for I/O automata. The auxiliary functions are defined in terms of the external and quiescent traces an automaton (or an expression) exhibits. In Chapter 5 we will provide related definitions in terms of the syntactic structure of the expressions.

We start by defining the set of *Weakly Specified Input actions* of an automaton:

$$Wsi(e) = \{a \in in(e) | \exists t \in ext(e)^* : at \notin qtraces(e)\}.$$

The idea behind the definition of *Wsi* is the following: if a specification of a device specifies something about the behavior of the device in the presence of an input action a, then not all choices of implementation should be correct when dealing with action a, i.e., some sequences of actions should not be allowed after performing action a. The word *Weakly* emphasizes the fact that we are abstracting from internal actions.

Another useful set is the set of *Weakly Specified Output actions* of an automaton:

$$Wso(e) = \{a \in out(e) | a \in etraces(e)\}.$$

$Wso(e)$ is the set of output actions that could become enabled according to the specification e. The word *Weakly* emphasizes the fact that we are considering output enabled actions up to internal transitions. In other words, as for *Wsi*, we are abstracting from internal actions. The usefulness of *Wso* is clear when stating distributivity of hiding over external choice. It is not true in general that $\tau_I(e\,{}_H{+}_K\,f) \equiv_Q \tau_I(e)\,{}_H{+}_K\,\tau_I(f)$ since performing an action from I resolves the choice context in the left automaton but does not resolve it in the right one. The condition for the above equivalence to hold turns out to be $Wso(e) \cap I = Wso(f) \cap I = \emptyset$.

Other useful functions are

$$Localen(e) = \{a \in local(e) | \exists e', e \xrightarrow{a} e'\},$$
$$Inten(e) = true \text{ iff } \tau \in Localen(e) \text{ and}$$
$$Quiet(e) = true \text{ iff } Localen(e) = \emptyset.$$

4.2 General theorems

In this subsection we present some general theorems that are sound for the quiescent preorder over I/O automata. We call them "theorems" since they are viewed as properties of I/O automata. Each expression stands for an I/O automaton and the operators are those of I/O automata. Moreover, the auxiliary functions are defined in terms of the external and quiescent traces of the considered automata. In the next chapter we will define some other syntactic functions to be substituted for the semantic ones and the theorems of this section will be called "axioms" by viewing the expressions as actual DIOA expressions and the operators as DIOA operators. Note that by the word "sound" we mean that the given theorems state valid properties of I/O automata. When dealing with axioms, instead, the word "sound" means that the relationship between two syntactic expressions stated by an axiom is valid in the Input/Output automaton model.

The first group of theorems concern the relationship between Ω and the other operators. In particular theorem **M** states that any automaton is an implementation of Ω.

Proposition 4.2.1 (omega theorems) *Let e be an I/O automaton. The following theorems are sound.*

R $\rho(\Omega_S) \equiv_Q \Omega_{\rho(S)}$

M $e \sqsubseteq_Q \Omega$

I $\tau_I(\Omega_S) \equiv_Q \Omega_{S'}$ *where* $S' = (in(S), out(S) \backslash I)$

P $\Omega_{S_1} \| \Omega_{S_2} \equiv_Q \Omega_{S_3}$ *where S_3 is the composition of S_1 and S_2*

■

The following theorems concern the renaming operator, which is distributive over every other operator.

Proposition 4.2.2 (renaming theorems) *Let e, f be I/O automata. The following theorems are sound.*

$\mathbf{R}_1$ $\rho(nil) \equiv_Q nil$

$\mathbf{R}_2$ $\rho(a \,.\, e) \equiv_Q \rho(a) \,.\, \rho(e)$

$\mathbf{R}_3$ $\rho(e \oplus f) \equiv_Q \rho(e) \oplus \rho(f)$

$\mathbf{R}_4$ $\rho(e \,{}_I{+}_J\, f) \equiv_Q \rho(e) \,{}_{\rho(I)}{+}_{\rho(J)}\, \rho(f)$

$\mathbf{R}_5$ $\rho_1(\rho_2(e)) \equiv_Q \rho_1 \circ \rho_2(e)$

$\mathbf{R}_6$ $\rho(\tau_I(e)) \equiv_Q \tau_{\rho'(I)}(\rho'(e))$ *if ρ' extends ρ*

$\mathbf{R}_7$ $\rho(e \| f) \equiv_Q \rho(e) \| \rho(f)$

■

The following theorems concern the parallel operator. This operator is commutative and associative, but does not have a neutral element. In fact in general $e \| nil \not\equiv_Q e$. The problem is that nil may have the control of some actions (essentially its output actions) which disappears by only considering e. However a weaker property is valid saying that two automata Ω can be collapsed (see theorem $\mathbf{P}$). Theorem $\mathbf{P}_3$ describes the properties of the parallel composition of an Ω automaton with a nil automaton.

Proposition 4.2.3 (parallel theorems) *Let e, f and g be I/O automata. The following theorems are sound.*

$\mathbf{P}_1$ $e \| f \equiv_Q f \| e$

$\mathbf{P}_2$ $(e \| f) \| g \equiv_Q e \| (f \| g)$

$\mathbf{P}_3$ $\Omega_{S_1} \| nil_{S_2} \sqsubseteq_Q \Omega_{S_3} \| nil_{S_4}$ *if* $(out(S_1) \subseteq out(S_3)) \wedge ((in(S_2) \subseteq in(S_4)) \vee out(S_4) = \emptyset)$

■

The following theorems concern the internal choice operator. Theorems $\mathbf{Ic}_{1,2,3}$ state commutativity, associativity and idempotence. Theorems $\mathbf{Ic}_{4,5,6,7}$ state the distributivity of all the operators of I/O automata (DIOA) over $\oplus$. Theorem $\mathbf{Ic}_8$ is immediate.

Proposition 4.2.4 (internal choice theorems) *Let e, f, g be I/O automata. The following theorems are sound.*

$\mathbf{Ic}_1$ $e \oplus f \equiv_Q f \oplus e$

$\mathbf{Ic}_2$ $(e \oplus f) \oplus g \equiv_Q e \oplus (f \oplus g)$

$\mathbf{Ic}_3$ $e \equiv_Q e \oplus e$

$\mathbf{Ic}_4$ $a\,.\,(e \oplus f) \equiv_Q a\,.\,e \oplus a\,.\,f$

$\mathbf{Ic}_5$ $(e \oplus f)\ {}_I{+}_J\ g \equiv_Q (e\ {}_I{+}_J\ g) \oplus (f\ {}_I{+}_J\ g)$

$\mathbf{Ic}_6$ $\tau_I(e \oplus f) \equiv_Q \tau_I(e) \oplus \tau_I(f)$

$\mathbf{Ic}_7$ $(e \oplus f) \| g \equiv_Q (e \| g) \oplus (f \| g)$

$\mathbf{Ic}_8$ $e \sqsubseteq_Q e \oplus f$

■

The following theorems concern the external choice operator. This is the most complicated operator of DIOA. The first two theorems state a sort of commutative and associative property. In fact they are not really commutative and associative properties since the operator changes. Theorem $\mathbf{Ec}_3$ states a sort of idempotence property. This property is not valid in general since, as noted in the introduction, the parameters of the choice operator play an important role. Theorem $\mathbf{Ec}_4$ permits duplicating an automaton e inside a choice context. Theorem $\mathbf{Ec}_4$ is different from theorem $\mathbf{Ec}_3$ in that the presence of parameter I does not require any condition on $Wsi(e)$.

Theorems $\mathbf{Ec}_{5,6,7,8}$ deal with the possibilities of adding or removing automata from a choice context. Their combinations give rise to theorems $\mathbf{Ec}_{15,16}$. Theorem $\mathbf{Ec}_7$ is particularly interesting since it expresses the main idea of our demonic approach: if e is not specifying anything about the occurrence of an input action a then any choice of implementation in the presence of a is correct.

Theorem $\mathbf{Ec}_9$ is a direct consequence of the definition of function Wsi. Its use, associated with theorems $\mathbf{Ec}_{5,7}$, gives rise to theorem $\mathbf{Ec}_{14}$. Theorem $\mathbf{Ec}_{14}$ permits to minimize the cardinality of the parameters of the external choice operator. Finally, theorems $\mathbf{Ec}_{10,11,12,13}$ state some relationships between the internal and external choice operators.

Proposition 4.2.5 (external choice theorems) *Let e, f, g be I/O automata. The following theorems are sound.*

$\mathbf{Ec}_1$ $e\ {}_I{+}_J\ f \equiv_Q f\ {}_J{+}_I\ e$

$\mathbf{Ec}_2$ $(e\ {}_I{+}_J\ f)\ {}_{I\cup J}{+}_K\ g \equiv_Q e\ {}_I{+}_{J\cup K}\ (f\ {}_J{+}_K\ g)$

$\mathbf{Ec}_3$ $e \equiv_Q e\ {}_I{+}_J\ e$ *if* $Wsi(e) \subseteq I \cup J$

$\mathbf{Ec}_4$ $e\ {}_I{+}_J\ f \equiv_Q (e\ {}_H{+}_K\ e)\ {}_I{+}_J\ f$ *if* $I \subseteq H \cup K$

$\mathbf{Ec}_5$ $\dfrac{(not(Quiet(e)) \wedge not(Inten(e))) \vee Quiet(f)}{e \sqsubseteq_Q e\ {}_I{+}_J\ f}$ *if* $J \cap Wsi(f) \subseteq I$

$\mathbf{Ec}_6$ $\dfrac{(not(Quiet(e)) \wedge not(Inten(e))) \vee Quiet(f)}{e\ {}_I{+}_J\ g \sqsubseteq_Q (e\ {}_H{+}_K\ f)\ {}_I{+}_J\ g}$ *if* $K \cap Wsi(f) \cap I \subseteq H$

$\mathbf{Ec}_7$ $\dfrac{Quiet(f)}{e\ {}_I{+}_J\ f \sqsubseteq_Q e}$ *if* $Wsi(e) \subseteq I$ *and* $Wsi(e) \cap J = \emptyset$

$\mathbf{Ec}_8$ $\dfrac{Quiet(f)}{(e\ {}_H{+}_K\ f)\ {}_I{+}_J\ g \sqsubseteq_Q e\ {}_I{+}_J\ g}$ *if* $Wsi(e) \cap I \subseteq H$ *and* $K \cap Wsi(e) \cap I = \emptyset$

$\mathbf{Ec}_9$ $e \equiv_Q e\ {}_I{+}_J\ a\,.\,\Omega$ *if* $Wsi(e) \subseteq I$ *and* $Wsi(e) \cap J = \emptyset$

$\mathbf{Ec}_{10}$ $a\,.\,e\ {}_I{+}_J\ a\,.\,f \equiv_Q a\,.\,(e \oplus f)$ *if* $a \in out(e) \cup (I \cap J)$

$\mathbf{Ec}_{11}$ $e\ {}_I{+}_J\ f \sqsubseteq_Q e \oplus f$ *where* $Wsi(e) \cap Wsi(f) \subseteq I \cup J$

$\mathbf{Ec}_{12}$ $\dfrac{Quiet(e) \Longleftrightarrow Quiet(f) \wedge not(Inten(e)) \wedge not(Inten(f))}{e\ {}_I{+}_J\ f \equiv_Q e \oplus f}$ *if* $Wsi(e) \cup Wsi(f) \subseteq I \cap J$

$\mathbf{Ec}_{13}$ $\dfrac{a \in in(e) \vee (not(Quiet(g)) \wedge not(Inten(g))) \vee Quiet(f)}{(a\,.\,e\ {}_I{+}_J\ f) \oplus g \equiv_Q (a\,.\,e\ {}_I{+}_J\ f) \oplus (a\,.\,e\ {}_I{+}_K\ g)}$ *if* $\begin{array}{l} Wsi(g) \subseteq K, \text{ and} \\ \{a\} \cap I \subseteq \{a\} \cap K \end{array}$

The following theorems are derived from the theorems above:

$\mathbf{Ec}_{14}$ $e\ {}_{I}{+}_{J}\ f \equiv_Q e\ {}_{I\setminus\{a\}}{+}_{J\setminus\{a\}}\ f$ *if* $a \in I \setminus Wsi(e)$.

$\mathbf{Ec}_{15}$ $\dfrac{Quiet(f)}{e \equiv_Q e\ {}_{I}{+}_{\emptyset}\ f}$ *where* $Wsi(e) \subseteq I$

$\mathbf{Ec}_{16}$ $\dfrac{Quiet(f)}{e\ {}_{I}{+}_{J}\ g \equiv_Q (e\ {}_{I}{+}_{K}\ f)\ {}_{I}{+}_{J}\ g}$ *if* $K \cap I = \emptyset$

Proof. We prove only theorem $\mathbf{Ec}_3$. Other examples of proofs are given for the hiding theorems. Due to Proposition 3.2.1 of chapter 3, the proof can be given by using the transition rules for DIOA. We also use a new notation $e \stackrel{a}{\Longrightarrow} e'$ meaning that there are two automata f, f' and two integers i, j such that $e \xrightarrow{\tau^i} f \xrightarrow{a} f' \xrightarrow{\tau^j} e'$.

Let t be an external (quiescent) trace of e. If $t = \lambda$ and t is quiescent, then, by definition of quiescent trace, there is a quiescent automaton e' such that $e \stackrel{\lambda}{\Longrightarrow} e'$. From rules $\mathbf{ech}_{4,5}$ $e\ {}_{I}{+}_{J}\ e \stackrel{\lambda}{\Longrightarrow} e'\ {}_{I}{+}_{J}\ e'$ which is quiescent. Therefore, λ is an external (quiescent) trace of $e\ {}_{I}{+}_{J}\ e$. If $t \neq \lambda$ then $t = at'$ for some external action a. In particular there is an automaton e' such that $e \stackrel{a}{\Longrightarrow} e'$ and t' is an external (quiescent) trace of e'. If $a \in I \cup J \cup out(e)$, then, from rules $\mathbf{ech}_{1,2}$, $e\ {}_{I}{+}_{J}\ e \stackrel{a}{\Longrightarrow} e'$, hence at' is an external (quiescent) trace of $e\ {}_{I}{+}_{J}\ e$. concluded; if $a \notin I \cup J \cup out(e)$ then, from rule $\mathbf{ech}_3$, $e\ {}_{I}{+}_{J}\ e \xrightarrow{a} \Omega$ and t is trivially an external (quiescent) trace of $e\ {}_{I}{+}_{J}\ e$ since any trace is a quiescent trace of Ω.

Conversely let t be an external (quiescent) trace of $e\ {}_{I}{+}_{J}\ e$. If $t = \lambda$ and t is quiescent, then, by definition of quiescent trace, there are two quiescent automata e', e'' such that $e\ {}_{I}{+}_{J}\ e \stackrel{\lambda}{\Longrightarrow} e'\ {}_{I}{+}_{J}\ e''$ where $e \stackrel{\lambda}{\Longrightarrow} e'$ and $e \stackrel{\lambda}{\Longrightarrow} e''$. The fact that λ is a quiescent trace of e is immediate from the hypothesis above. If $t \neq \lambda$ then $t = at'$ for some action a. If $a \in I \cup J \cup out(e)$, then, from rules $\mathbf{ech}_{1,2}$, there is an automaton e' such that $e\ {}_{I}{+}_{J}\ e \stackrel{a}{\Longrightarrow} e'$ where $e \stackrel{a}{\Longrightarrow} e'$ and t' is an external (quiescent) trace of e'. The conclusion is immediate once again. If $a \notin I \cup J \cup out(e)$, then a is an input action and $a \notin Wsi(e)$ since $Wsi(e) \subseteq I \cup J$. From the definition of Wsi, at' is an external (quiescent) trace of e, hence the proof is concluded. ■

The following theorems concern the hiding operator. The first seven theorems show the relations between the hiding operator and the other ones. In particular theorem $\mathbf{I}_4$ establishes the distributivity of hiding over choice (this is the place where function Wso is used); theorem $\mathbf{I}_7$ is simply a way of saying that internal actions can be renamed. Theorems $\mathbf{I}_{8,9}$ state some ways of dealing with the hiding operator when it does not distribute over prefixing or external choice.

The rest of the theorems permit eliminating/adding internal actions from automata. Theorem $\mathbf{I}_{10}$ essentially says that $\tau_I(e)$ is an implementation of $\tau_I(\tau_1.e)$. In fact the second automaton can move to the unspecified state with every input action before performing action τ_1 while the first process may not. The condition for which the two automata can be considered equivalent is when also $\tau_I(e)$ can perform any trace after any input action. A sufficient condition is then $Wsi(e) = \emptyset$ and this is what is stated in theorem $\mathbf{I}_{11}$.

Theorems $\mathbf{I}_{12,13}$ permit eliminating explicit internal actions, possibly by transforming an external choice into an internal one. Theorems $\mathbf{I}_{14,15}$ permit eliminating the hiding operator from particular classes of I/O automata that are expressible through DIOA expressions. These theorems are particular important in their axiom version to achieve completeness.

Theorems $\mathbf{I}_{16,17}$ are derived from the above theorems and are useful for the applications. Theorem $\mathbf{I}_{16}$ eliminates internal actions interleaved with an external one. Note that, by using the external choice theorems together with theorems $\mathbf{I}_{11,12,13}$, the statement of theorem $\mathbf{I}_{16}$ can be generalized to the case in which there is any number of hidden actions interleaved with a.

Theorem $\mathbf{I}_{17}$ says that, if the effect of a prefix with an internal action is simply to temporary block a process that can perform only locally controlled actions, then the prefix can be removed and the automaton can be simplified. It is a consequence of theorems $\mathbf{I}_{13}$ and $\mathbf{Ec}_{1,2,4}$.

Proposition 4.2.6 (hiding theorems) *Let e, f, g be I/O automata and let $i \in I$. The following theorems are sound.*

$\mathbf{I}_1$ $\tau_\emptyset(e) \equiv_Q e$

$\mathbf{I}_2$ $\tau_I(nil) \equiv_Q nil$

$\mathbf{I}_3$ $\tau_I(a \,.\, e) \equiv_Q a \,.\, \tau_I(e)$ *if* $a \notin I$

$\mathbf{I}_4$ $\tau_I(e \;{}_H{+}_K\; f) \equiv_Q \tau_I(e) \;{}_H{+}_K\; \tau_I(f)$ *if* $Wso(e) \cap I = Wso(f) \cap I = \emptyset$

$\mathbf{I}_5$ $\tau_I(\tau_J(e)) \equiv_Q \tau_{I \cup J}(e)$

$\mathbf{I}_6$ $\tau_I(e) \| \tau_J(f) \equiv_Q \tau_{I \cup J}(e \| f)$ *if* $I \cap acts(f) = J \cap acts(e) = \emptyset$

$\mathbf{I}_7$ $e \equiv_Q \rho(e)$ *if* ρ *is the identity function*

$\mathbf{I}_8$ $\dfrac{\tau_I(e) \sqsubseteq_Q \tau_I(f)}{\tau_I(a\,.\,e) \sqsubseteq_Q \tau_I(a\,.\,f)}$

$\mathbf{I}_9$ $\dfrac{\tau_I(e) \sqsubseteq_Q \tau_I(g)}{\tau_I(e\ {}_H{+}_K\ f) \sqsubseteq_Q \tau_I(g\ {}_H{+}_K\ f)}$

$\mathbf{I}_{10}$ $\tau_I(e) \sqsubseteq_Q \tau_I(i\,.\,e\ {}_H{+}_K\ f)$

$\mathbf{I}_{11}$ $\tau_I(i\,.\,e) \equiv_Q \tau_I(e)$ *if* $Wsi(e) = \emptyset$

$\mathbf{I}_{12}$ $\dfrac{not(Quiet(e)) \wedge not(Inten(e))}{\tau_I(e\ {}_H{+}_\emptyset\ i\,.\,f) \equiv_Q \tau_I(e \oplus f)}$ *if* $Wsi(e) \subseteq H$

$\mathbf{I}_{13}$ $\dfrac{Quiet(e)}{\tau_I(e\ {}_H{+}_\emptyset\ i\,.\,f) \equiv_Q \tau_I(e\ {}_K{+}_K\ f)}$ *if* $Wsi(e) \subseteq H$ *and* $Wsi(e) \subseteq K$

$\mathbf{I}_{14}$ $\tau_I((\Omega_{S_0} \| nil_{S_1} \| \cdots \| nil_{S_n}) \| e) \equiv_Q \tau_I(\Omega \| e)$ *if* $\forall_{1 \le j \le n} (out(S_0) \cap in(S_j) \cap I) \backslash in(e) \neq \emptyset$

$\mathbf{I}_{15}$ $\tau_I(\Omega_{S_0} \| nil_{S_1} \| \cdots \| nil_{S_n}) \equiv_Q \Omega_{S_0 \backslash I} \| nil_{S_1 \backslash I} \| \cdots \| nil_{S_n \backslash I}$ *if* $\forall_{1 \le i \le n} out(S_0) \cap in(S_i) \cap I = \emptyset$

The following theorems are derived from the theorems above:

$\mathbf{I}_{16}$ $\tau_I(a\,.\,i\,.\,e\ {}_{\{a\} \cap in(e)}{+}_\emptyset\ i\,.\,a\,.\,e) \equiv_Q \tau_I(a\,.\,e)$ *if* $Wsi(e) = \emptyset$

$\mathbf{I}_{17}$ $\tau_I(i\,.\,(e\ {}_\emptyset{+}_J\ f)\ {}_\emptyset{+}_J\ f) \equiv_Q \tau_I(e\ {}_\emptyset{+}_J\ f)$ *if* $Quiet(f)$ *and* $Wsi(f) \subseteq J$

Proof. We only prove theorems $\mathbf{I}_{12,13,14,15}$. The other theorems are proven in the same way.

$\mathbf{I}_{12}$ Let t be an external (quiescent) trace of $\tau_I(e_H{+}_\emptyset i.f)$. By the transition rules for τ_I and the definition of external trace, there is a trace t' of $e_H{+}_\emptyset i.f$ such that $t' \lceil ext(\tau_I(e_H{+}_\emptyset i.f)) = t$ and t' leads the system to a quiescent state if t is quiescent. Note that, since $\tau_I(e\ {}_H{+}_\emptyset\ i\,.\,f)$ is not quiescent, $t' \neq \lambda$ if $t = \lambda$ and t is quiescent. Since no internal actions are enabled from e then the first action of t' is not τ and rules $\mathbf{ech}_{4,5}$ are not used for the first transition of t'. We distinguish the following cases:

(a) rule $\mathbf{ech}_1$ is used for the first transition of t'

In this case $e\ {}_H{+}_\emptyset\ i\,.\,f \xrightarrow{a} e'$ for some action a where $e \xrightarrow{a} e'$. By rule $\mathbf{ich}_1$ $e \oplus f \xrightarrow{\tau} e \xrightarrow{a} e'$, hence t is trivially an external (quiescent) trace of $\tau_I(e \oplus f)$.

(b) rule $\mathbf{ech}_2$ is used for the first transition of t'

In this case $e\ _H\!+_\emptyset\ i\,.\,f \xrightarrow{i} f$ and $\tau_I(e\ _H\!+_\emptyset\ i\,.\,f) \xrightarrow{\tau} \tau_I(f)$. By rules $\mathbf{ich}_1$ and $\mathbf{tau}_1$ $\tau_I(e \oplus f) \xrightarrow{\tau} \tau_I(f)$ and the conclusion is immediate again.

(c) rule $\mathbf{ech}_3$ is used for the first transition of t'

In this case $e\ _H\!+_\emptyset\ i\,.\,f \xrightarrow{a} \Omega$ for some input action $a \notin H$. In particular $t = at''$ for some trace t'' and, since $H \subseteq Wsi(e)$, $a \notin Wsi(e)$. From the definition of Wsi we have that at'' is an external (quiescent) trace of e, hence at'' is an external (quiescent) trace of $\tau_I(e \oplus f)$.

A similar and simpler argument shows the converse trace inclusion.

$\mathbf{I}_{13}$ Let t be an external (quiescent) trace of $\tau_I(e\ _H\!+_\emptyset\ i\,.\,f)$. If $t = \lambda$ and t is a quiescent trace, then, since e is quiescent and $i\,.\,f$ is not quiescent, it must be $\tau_I(e\ _H\!+_\emptyset\ i\,.\,f) \xrightarrow{\tau} \tau_I(f) \xRightarrow{\lambda} \tau_I(f')$ where f' is quiescent. On the other side $\tau_I(e\ _K\!+_K\ f) \xRightarrow{\lambda} \tau_I(g)$ where either $g \equiv e\ _K\!+_K\ f'$ or $g \equiv f'$ depending on the trace leading to f'. Since e is quiescent, then in both cases g is quiescent and λ is a quiescent trace of $\tau_I(e\ _K\!+_K\ f)$. Suppose now that $t \neq \lambda$. By the transition rules for τ_I and the definition of external trace, there is a trace t' of $e\ _H\!+_\emptyset\ i\,.\,f$ such that $t'\lceil ext(\tau_I(e\ _H\!+_\emptyset\ i\,.\,f)) = t$ and t' leads the system to a quiescent state if t is quiescent. Since no internal actions are enabled from e, then the first action of t' is not τ and rules $\mathbf{ech}_{4,5}$ are not used for the first transition of t'. We distinguish the following cases:

(a) rule $\mathbf{ech}_1$ is used for the first transition

In this case $e\ _H\!+_\emptyset\ i\,.\,f \xrightarrow{a} e'$ for some action a where $e \xrightarrow{a} e'$ and $a \in H \cup out(e)$. If $a \in K \cup out(e)$ then rule $\mathbf{ech}_1$ is applicable to $e\ _K\!+_K\ f$ leading the right automaton to $\tau_I(e')$. The conclusion is then immediate. If $a \notin K \cup out(e)$ then rule $\mathbf{ech}_3$ is applicable to $e\ _K\!+_K\ f$ leading the system to Ω. The conclusion is immediate again.

(b) rule $\mathbf{ech}_2$ is used for the first transition

In this case $e\ _H\!+_\emptyset\ i\,.\,f \xrightarrow{i} f$ and $\tau_I(e\ _H\!+_\emptyset\ i\,.\,f) \xrightarrow{\tau} \tau_I(f)$. Let $t' = i\tau^n bt''$. Since $\tau^n bt''$ is a trace of f, we have that $\exists f', f'' | f \xrightarrow{\tau^{n-1}} f' \xrightarrow{b} f''$ where t'' is a trace of f'' leading the system to a quiescent state if t is quiescent. By the transition rules for the external choice operator, $e\ {}_K\!+_K\ f \xrightarrow{\tau^n} e\ {}_K\!+_K\ f' \xrightarrow{b} g$ where g is either f'' or Ω depending on the rule used for the b-transition ($\mathbf{ech}_2$ or $\mathbf{ech}_3$). In both the cases t'' is a trace of g leading the system to a quiescent state if t is quiescent. The conclusion is then immediate.

(c) rule $\mathbf{ech}_3$ is used for the first transition

In this case $e\ {}_H\!+_\emptyset\ i\ .\ f \xrightarrow{a} \Omega$ for some input action a. In particular $a \notin Wsi(e)$, hence rule $\mathbf{ech}_3$ is also applicable to $e\ {}_K\!+_K\ f$ leading the right automaton to Ω. The conclusion is then immediate.

A similar and simpler argument shows the converse trace inclusion.

$\mathbf{I}_{14}$ For each $1 \leq i \leq n$ choose $a_i \in (out(S_0) \cap in(S_j) \cap I) \backslash in(e)$. Then

$$(\Omega_{S_0} \| nil_{S_1} \| \cdots \| nil_{S_n}) \| e \xrightarrow{a_1 \cdots a_n} (\Omega_{S_0} \| \Omega_{S_1} \| \cdots \| \Omega_{S_n}) \| e \text{ and}$$

$$\tau_I((\Omega_{S_0} \| nil_{S_1} \| \cdots \| nil_{S_n}) \| e) \overset{\lambda}{\Longrightarrow} \tau_I((\Omega_{S_0} \| \Omega_{S_1} \| \cdots \| \Omega_{S_n}) \| e)$$

which, by axiom P, is equivalent to $\tau_I(\Omega \| e)$, hence

$$\tau_I(\Omega \| e) \sqsubseteq_Q \tau_I((\Omega_{S_0} \| nil_{S_1} \| \cdots \| nil_{S_n}) \| e).$$

The other inclusion is trivial since each process is less than Ω (use theorem $\mathbf{M}$ and the substitutivity rules).

$\mathbf{I}_{15}$ Let t be an external (quiescent) trace of $\tau_I(\Omega_{S_0} \| nil_{S_1} \| \cdots \| nil_{S_n})$. We show by induction on the length of t that t is an external (quiescent) trace of $\Omega_{S_0 \backslash I} \| nil_{S_1 \backslash I} \| \cdots \| nil_{S_n \backslash I}$. If $t = \lambda$ then the result is immediate since λ is a quiescent trace of any automaton of the form $\Omega \| nil \| \cdots \| nil$. If $t \neq \lambda$ then $t = at'$ for some external action a. By the definition of external trace and the transition rules for τ_I, we have that $\Omega_{S_0} \| nil_{S_1} \| \cdots \| nil_{S_n} \xrightarrow{t_1} e \xrightarrow{a} e'$ for some e, e', t_1 where t_1 has actions in $I \cup \{\tau\}$. Since $\forall_{1 \leq i \leq n} out(S_0) \cap in(S_i) \cap I = \emptyset$, then $e \equiv f \| nil_{S_1} \| \cdots \| nil_{S_n}$ where f is either Ω_{S_0} or nil_{S_0}. In the case f is nil_{S_0} we have that $\Omega_{S_0\setminus I}\|nil_{S_1\setminus I}\|\cdots\|nil_{S_n\setminus I} \xrightarrow{\tau} nil_{S_0\setminus I}\|nil_{S_1\setminus I}\|\cdots\|nil_{S_n\setminus I}$ using rule $\mathbf{ome}_2$. Let

$$g = \begin{cases} \Omega_{S_0\setminus I}\|nil_{S_1\setminus I}\|\cdots\|nil_{S_n\setminus I} & \text{if } f \equiv \Omega_{S_0} \\ nil_{S_0\setminus I}\|nil_{S_1\setminus I}\|\cdots\|nil_{S_n\setminus I} & \text{if } f \equiv nil_{S_0} \end{cases}$$

In the transition $e \xrightarrow{a} e'$ there is a set of automata $\{nil_{S_{i_j}} : j \in J\}$ of $f\|nil_{S_1}\|\cdots\|nil_{S_n}$, having a is an input action, that will move to Ω. The set of automata $\{nil_{S_{i_j}\setminus I} : j \in J\}$ also move to Ω with action a on g since they all have action a as an input action. To conclude it is enough to collapse all Ω automata by repeatedly applying axiom P, and successively apply the induction hypothesis.

The inverse trace inclusion is easier to prove since each trace of $\Omega_{S_0\setminus I}\|nil_{S_1\setminus I}\|\cdots\|nil_{S_n\setminus I}$ has no actions from I.

■

4.3 Theorems for recursively defined processes

In this subsection we present some tools to deal with recursion by stating some properties about recursive definitions. We first find a class of recursive DIOA equations having unique solutions up to quiescent trace equivalence, i.e., a unique fixpoint; then, on the same class of equations, we state some properties of their pre and post fixpoints.

We consider the class of equations given by means of strongly guarded expressions (see Definition 4.3.2), i.e., expressions in which each process variable occurs within the scope of some not hidden prefix. For this class we can assure that every set of mutually recursive equations has a unique fixpoint. It is immediate to see that this property is not valid if we consider non-strongly guarded equations. Consider for example

$$X \stackrel{\text{def}}{=} \tau_{\{a\}}(a\,.\,(X\|nil))$$

where nil has a single output action a and $a \notin acts(X)$. Then every automaton with the same action signature as X is a solution of the equation.

Since recursion is expressed through DIOA expressions, we can interchangeably talk of expressions or talk of represented automata. Moreover we can interchangeably talk of transition rules applied to expressions or transition rules applied to automata. The only point in which it is not possible to talk about expressions is when some automata are substituted for the variables of a set of equations. We first introduce some notational conventions. We indicate with $\tilde{E}$ a set of expressions $\{E_1, \ldots, E_n\}$. The same convention is valid for process variables and for automata. With the notation $E[\tilde{P}/\tilde{X}]$ we mean the automaton obtained from E by simultaneously substituting all its occurrences of X_i with P_i for every i. With the notation $\tilde{E}[\tilde{P}/\tilde{X}]$ we mean the substitution above repeated for every expression E_i of $\tilde{E}$.

We now introduce the notion of strongly guarded expression, which is then generalized to a set of equations.

Definition 4.3.1 (strong guardedness) Given a set of actions A,

- nil is strongly guarded with respect to A,
- $a \,.\, e$ is strongly guarded with respect to A iff $a \notin A$ or e is strongly guarded with respect to A,
- $e_1 \oplus e_2$ is strongly guarded with respect to A iff both e_1 and e_2 are strongly guarded with respect to A,
- $e_1 \; {}_I\!+_J \; e_2$ is strongly guarded with respect to A iff both e_1 and e_2 are strongly guarded with respect to A,
- $\tau_I(e)$ is strongly guarded with respect to A iff e is strongly guarded with respect to $A \cup I$,
- $\rho(e)$ is strongly guarded with respect to A iff e is strongly guarded with respect to $\rho^{-1}(A)$, and
- $e_1 \| e_2$ is strongly guarded with respect to A iff both e_1 and e_2 are strongly guarded with respect to A.

A DIOA expression e is *strongly guarded* iff it is strongly guarded with respect to $\emptyset$. ■

Informally a DIOA expression e is strongly guarded with respect to a set of actions A iff every process variable of e occurs in a subexpression of the form $b \,.\, e'$ of e where b is an external action of e' that is transformed (renamed) into an external action of e not belonging to A. The use of parameter A is due to the presence of the hiding operator. The intuitive idea behind a strongly guarded expression e is that no process variable affects any transition from e. The following definition extends the concept of strong guardedness to a generic set of equations.

Definition 4.3.2 (strongly guarded equations) Given a set of equations $\tilde{X} \stackrel{\text{def}}{=} \tilde{E}(\tilde{X})$, an equation $X_k \stackrel{\text{def}}{=} E_k(\tilde{X})$ is *strongly guarded* with respect to A if $\exists A_1, \ldots, A_n$ such that

1. $\forall_i \, E_i(\tilde{X})$ is strongly guarded with respect to A_i,
2. $A \subseteq A_k$ and
3. for each X_j occurring within E_i, $A_i \cup A' \subseteq A_j$ where A' is the set of actions of X_j that are hidden within E_i.

$\tilde{X} \stackrel{\text{def}}{=} \tilde{E}(\tilde{X})$ is *strongly guarded* if, for each i, $X_i \stackrel{\text{def}}{=} E_i(\tilde{X})$ is strongly guarded with respect to $\emptyset$. ■

We can now state the main theorem of this section. As a corollary we have uniqueness of fixpoint for strongly guarded equations.

Theorem 4.3.3 (recursive substitutivity) *Let $\tilde{X} \stackrel{\text{def}}{=} \tilde{E}(\tilde{X})$ be a strongly guarded set of equations and let $\tilde{P}$ be a set of I/O automata. Then the following facts hold:*

1. *if $\tilde{P} \sqsubseteq_Q \tilde{E}[\tilde{P}/\tilde{X}]$ then $\tilde{P} \sqsubseteq_Q \tilde{Aut}(\tilde{X})$;*
2. *if $\tilde{E}[\tilde{P}/\tilde{X}] \sqsubseteq_Q \tilde{P}$ then $\tilde{Aut}(\tilde{X}) \sqsubseteq_Q \tilde{P}$.*

■

Corollary 4.3.4 (unique solution of equations) *Let $\tilde{X} \stackrel{\text{def}}{=} \tilde{E}(\tilde{X})$ be a strongly guarded set of equations and let $\tilde{P} \equiv_Q \tilde{E}[\tilde{P}/\tilde{X}]$ where $\tilde{P}$ is a set of automata.. Then $\tilde{P} \equiv_Q \tilde{Aut}(\tilde{X})$.*

Proof. Direct consequence of theorem 4.3.3. ■

The rest of this section is dedicated to the proof of theorem 4.3.3. The main idea of the proof is that, by unfolding a set of equations n times, every trace of length at most n can be generated independently of the automata substituted for the variables $\tilde{X}$. The first lemma formally introduces the unfoldings of the equations and proves some properties that will be fundamental to allow the above idea to work.

Lemma 4.3.5 (unfoldings) *Given a set of process variables $\tilde{X}$ consider the corresponding defining expressions $\tilde{E}(\tilde{X})$. Let $\tilde{E}^0 = \tilde{E}(\tilde{X})$ and, for each $n \geq 1$, $\tilde{E}^n = \tilde{E}[\tilde{E}^{n-1}/\tilde{X}]$. Let $\tilde{P}$ be a set of I/O automata. Then the following holds:*

1. *$\tilde{X} \equiv_Q \tilde{E}^n$ for each n.*

2. *$\tilde{P} \sqsubseteq_Q \tilde{E}[\tilde{P}/\tilde{X}] \Longrightarrow \tilde{P} \sqsubseteq_Q \tilde{E}^n[\tilde{P}/\tilde{X}]$ for each n.*

3. *$\tilde{E}[\tilde{P}/\tilde{X}] \sqsubseteq_Q \tilde{P} \Longrightarrow \tilde{E}^n[\tilde{P}/\tilde{X}] \sqsubseteq_Q \tilde{P}$ for each n.*

Proof.

1. By induction on n. If $n = 0$ then the result is immediate from the fact that $X \equiv_Q E(X)$ for each process variable X. Suppose by induction that $\tilde{X} \equiv_Q \tilde{E}^n$. By substitutivity, $\tilde{E}[\tilde{X}/\tilde{X}] \equiv_Q \tilde{E}[\tilde{E}^n/\tilde{X}]$. Since, by the base case, $\tilde{E}[\tilde{X}/\tilde{X}] \equiv_Q \tilde{X}$ and since, by definition, $\tilde{E}[\tilde{E}^n/\tilde{X}]$ is $\tilde{E}^{n+1}$, we can conclude that $\tilde{X} \equiv_Q \tilde{E}^{n+1}$.

2. By induction on n. If $n = 0$ then the assertion is true by definition. Suppose by induction that $\tilde{P} \sqsubseteq_Q \tilde{E}^n[\tilde{P}/\tilde{X}]$. By substitutivity, $\tilde{E}[\tilde{P}/\tilde{X}] \sqsubseteq_Q \tilde{E}[\tilde{E}^n[\tilde{P}/\tilde{X}]/\tilde{X}]$. Since by hypothesis $\tilde{P} \sqsubseteq_Q \tilde{E}[\tilde{P}/\tilde{X}]$ and since, by definition, $\tilde{E}[\tilde{E}^n[\tilde{P}/\tilde{X}]/\tilde{X}]$ is $\tilde{E}^{n+1}[\tilde{P}/\tilde{X}]$, we can conclude that $\tilde{P} \sqsubseteq_Q \tilde{E}^{n+1}[\tilde{P}/\tilde{X}]$.

3. By induction on n. If $n = 0$ then the assertion is true by definition. Suppose by induction that $\tilde{E}^n[\tilde{P}/\tilde{X}] \sqsubseteq_Q \tilde{P}$. By substitutivity, $\tilde{E}[\tilde{E}^n[\tilde{P}/\tilde{X}]/\tilde{X}] \sqsubseteq_Q \tilde{E}[\tilde{P}/\tilde{X}]$. Since by hypothesis $\tilde{E}[\tilde{P}/\tilde{X}] \sqsubseteq_Q \tilde{P}$ and since, by definition, $\tilde{E}[\tilde{E}^n[\tilde{P}/\tilde{X}]/\tilde{X}]$ is $\tilde{E}^{n+1}[\tilde{P}/\tilde{X}]$, we can conclude that $\tilde{E}^{n+1}[\tilde{P}/\tilde{X}] \sqsubseteq_Q \tilde{P}$.

■

The following lemmas essentially state the independence of the traces of length at most n from the automata substituted for the variables of $\tilde{E}^n$.

Lemma 4.3.6 *Let $E(\tilde{X})$ be strongly guarded and let $E(\tilde{X}) \xrightarrow{\tau} E'(\tilde{X})$. Then*

1. $E'(\tilde{X})$ is strongly guarded and

2. for each set of automata $\tilde{P}$, $E[\tilde{P}/\tilde{X}] \xrightarrow{\tau} E'[\tilde{P}/\tilde{X}]$.

Proof. We prove a more general result: Let $E(\tilde{X})$ be strongly guarded with respect to A and let $E(\tilde{X}) \xrightarrow{\alpha} E'(\tilde{X})$ where $\alpha \in A \cup \{\tau\}$. Then

1. $E'(\tilde{X})$ is strongly guarded with respect to A and

2. for each set of automata $\tilde{P}$, $E[\tilde{P}/\tilde{X}] \xrightarrow{\alpha} E'[\tilde{P}/\tilde{X}]$.

The lemma follows by taking $A = \emptyset$. We proceed by induction on the structure of E. If $E \equiv \mathit{nil}$ or $E \equiv \Omega$ then the result is trivial since no variables are contained in E. The result is trivial also when E is a process variable since E is not strongly guarded. For the induction step we consider cases depending on the most external operator.

Case 1 prefixing

Let $E \equiv a \,.\, E_1$. If $\alpha \neq a$ then the result is trivial since the only admitted transitions with action α from E move the system to Ω. If $\alpha = a$ then the transition is $a \,.\, E_1 \xrightarrow{a} E_1$ and, since $a \in A$, E_1 is strongly guarded with respect to A. Moreover $a \,.\, E'[\tilde{P}/\tilde{X}] \xrightarrow{a} E_1[\tilde{P}/\tilde{X}]$ for each set of automata $\tilde{P}$.

Case 2 choice

Let $E \equiv E_1 \;{}_I{+}_J\; E_2$. By definition of strong guardedness both E_1 and E_2 are strongly guarded with respect to A. For transitions to Ω the result is immediate; for transitions involving E_1 or E_2 the result follows directly from the induction hypothesis.

Case 3 hiding

Let $E \equiv \tau_I(E_1)$. By definition of strong guardedness E_1 is strongly guarded with respect to $A \cup I$. If $\tau_I(E_1) \xrightarrow{\alpha} \tau_I(E')$ where $\alpha \in A \cup \{\tau\}$ then, by the transition rules, $E_1 \xrightarrow{\beta} E'$ where $\beta \in A \cup I \cup \{\tau\}$. By induction E' is strongly guarded with respect to $A \cup I$ and $E_1[\tilde{P}/\tilde{X}] \xrightarrow{\beta} E'[\tilde{P}/\tilde{X}]$ for each set of automata $\tilde{P}$. In particular $\tau_I(E')$ is strongly guarded with respect to A and $\tau_I(E_1[\tilde{P}/\tilde{X}]) \xrightarrow{\alpha} \tau_I(E'[\tilde{P}/\tilde{X}])$.

Case 4 renaming

Let $E \equiv \rho(E_1)$. By definition of strong guardedness E_1 is strongly guarded with respect to $\rho^{-1}(A)$. If $\rho(E_1) \xrightarrow{\alpha} \rho(E')$ where $a \in A \cup \{\tau\}$ then, by the transition rules, $E_1 \xrightarrow{\rho^{-1}(\alpha)} E'$ where $\rho^{-1}(\alpha) \in \rho^{-1}(A) \cup \{\tau\}$. By induction E' is strongly guarded with respect to $\rho^{-1}(A)$ and $E_1[\tilde{P}/\tilde{X}] \xrightarrow{\rho^{-1}(\alpha)} E'[\tilde{P}/\tilde{X}]$ for each set of automata $\tilde{P}$. In particular $\rho(E')$ is strongly guarded with respect to A and $\rho(E_1[\tilde{P}/\tilde{X}]) \xrightarrow{\alpha} \rho(E'[\tilde{P}/\tilde{X}])$.

Case 5 parallel

Let $E \equiv E_1 \| E_2$. By definition of strong guardedness both E_1 and E_2 are strongly guarded with respect to A. It is enough to apply the induction hypothesis to E_1 and E_2 to conclude.

■

Lemma 4.3.7 *Let $E(\tilde{X})$ be strongly guarded and let $E(\tilde{X}) \xrightarrow{\tau^n} E'(\tilde{X})$. Then*

1. *$E'(\tilde{X})$ is strongly guarded and*

2. *for each set of automata $\tilde{P}$, $E[\tilde{P}/\tilde{X}] \xrightarrow{\tau^n} E'[\tilde{P}/\tilde{X}]$.*

Proof. By induction on n. If $n = 0$ then the result is trivial. Suppose now that the fact is valid for n and let $E(\tilde{X}) \xrightarrow{\tau\tau^n} E'(\tilde{X})$. By means of Lemma 4.3.6 we perform the first step and, by induction, we perform the remaining n steps. ■

To state the following lemmas we need a definition.

Definition 4.3.8 (transitional equivalence between I/O automata) Two I/O automata A, B are *transitional equivalent* ($A \equiv B$) iff their transition trees are isomorphic, i.e., there is an isomorphism h from the reachable states of A to the reachable states of B such that for each reachable $q \in states(A)$, $q \xrightarrow{a} q'$ iff $h(q) \xrightarrow{a} h(q')$. ■

In the following lemmas we use the transition rules for DIOA in order to derive the transitions of an automaton.

Lemma 4.3.9 *Let $E(\tilde{X})$ be strongly guarded and let $\tilde{P}$ be a set of automata. Let $E[\tilde{P}/\tilde{X}] \xrightarrow{\tau^n} O$. Then $\exists E'' : E(\tilde{X}) \xrightarrow{h} E''(\tilde{X})$ and $O \equiv E''[\tilde{P}/\tilde{X}]$.*

Proof. The proof method is exactly the same as the one used in lemmas 4.3.6 and 4.3.7. Note that the lemma is valid also when $\tilde{P}$ are expressions. ∎

Lemma 4.3.10 *Let $E(\tilde{X})$ be strongly guarded and let $E(\tilde{X}) \xrightarrow{a} E'(\tilde{X})$. Then, for each set of automata $\tilde{P}$, $E[\tilde{P}/\tilde{X}] \xrightarrow{a} E'[\tilde{P}/\tilde{X}]$.*

Proof. The proof method is exactly the same as in Lemma 4.3.6. ∎

Lemma 4.3.11 *Let $E(\tilde{X})$ be strongly guarded and let $\tilde{P}$ be a set of automata. Let $E[\tilde{P}/\tilde{X}] \xrightarrow{a} O$. Then $\exists E'' : E(\tilde{X}) \xrightarrow{a} E''(\tilde{X})$ and $O \equiv E''[\tilde{P}/\tilde{X}]$.*

Proof. The proof method is exactly the same as in Lemma 4.3.6. Note that the lemma is valid also when $\tilde{P}$ are expressions. ∎

Lemma 4.3.12 *Let $E(\tilde{X})$ be strongly guarded. Then $\lambda \in qtraces(E[\tilde{P}/\tilde{X}])$ iff $\lambda \in qtraces(E)$.*

Proof. Suppose $\lambda \in qtraces(E[\tilde{P}/\tilde{X}])$. By definition $E[\tilde{P}/\tilde{X}] \xrightarrow{\tau^n} O$ for some $n \geq 0$ where O is quiescent. By Lemma 4.3.9 $\exists E'' : E(\tilde{X}) \xrightarrow{\tau^n} E''(\tilde{X})$ and $E' \equiv E''[\tilde{P}/\tilde{X}]$. Suppose E'' not to be quiescent. Then $E''[\tilde{X}] \xrightarrow{o} E'''$ for some local action o. By Lemmas 4.3.10 and 4.3.11 there is a transition from O with action o. This gives a contradiction, hence E'' is quiescent and $\lambda \in qtraces(E)$. The converse is analogous. ∎

Before stating the main lemma we need a new definition.

Definition 4.3.13 Let $F(\tilde{Y})$ be a DIOA expression with k variables, and $\tilde{X} \stackrel{\text{def}}{=} E(\tilde{X})$ be a strongly guarded set of k equations. F is said *strongly compatible* with $\tilde{E}$ if, for each Y_i occurring within F, $X_i \stackrel{\text{def}}{=} E_i(\tilde{X})$ is strongly guarded with respect to A where A is the set of actions of Y_i that are hidden in F from the considered occurrence of Y_i. ∎

Lemma 4.3.14 *Let $F(\tilde{Y})$ be a DIOA expression with k variables, and let $\tilde{X} \stackrel{\text{def}}{=} E(\tilde{X})$ be a strongly guarded set of k equations where F is strongly compatible with $\tilde{E}$. Then*

1. *$F[\tilde{E}/\tilde{Y}]$ is strongly guarded;*

2. *if F is strongly guarded and $F[\tilde{X}] \xrightarrow{\alpha} F'$ (where α could be τ), then F' is strongly compatible with $\tilde{E}$.*

Proof. Item 1 follows from the definitions of strong guardedness and strong compatibility; the proof of item 2 is by induction and follows the same lines of Lemma 4.3.6. ∎

We can now prove the main lemma which relates the automata $\tilde{X}$ to the automata substituted for the variables. Note that lemma 4.3.5 plays an essential role in this proof. The introduction of F is necessary to set up an inductive process.

Lemma 4.3.15 *Let $F(\tilde{Y})$ be an expression with k variables, $\tilde{P}$ be a set of k automata, and $\tilde{X} \stackrel{\mathrm{def}}{=} E(\tilde{X})$ be a strongly guarded set of k equations where F is strongly compatible with $\tilde{E}$ and the variables of $\tilde{X}$ are disjoint from those of $\tilde{Y}$. Let h be a trace of length n. Then h is an external (quiescent) trace of $F[\tilde{E}^n[\tilde{P}/\tilde{X}]/\tilde{Y}]$ iff h is an external (quiescent) trace of $F[\tilde{E}^n/\tilde{Y}]$.*

Proof. We prove both directions by induction on n. We also use the following syntactical identities:

1. $F[\tilde{E}[\tilde{P}/\tilde{X}]/\tilde{Y}] = F[\tilde{E}/\tilde{Y}][\tilde{P}/\tilde{X}]$.

2. $F[\tilde{E}^{n+1}[\tilde{P}/\tilde{X}]/\tilde{Y}] = F[\tilde{E}/\tilde{Y}][\tilde{E}^n[\tilde{P}/\tilde{X}]/\tilde{X}]$.

($\Rightarrow$) Suppose that λ is an external (quiescent) trace of $F[\tilde{E}[\tilde{P}/\tilde{X}]/\tilde{Y}]$. From identity *1*, λ is an external (quiescent) trace of $F[\tilde{E}/\tilde{Y}][\tilde{P}/\tilde{X}]$. By Lemma 4.3.14, $F[\tilde{E}/\tilde{Y}]$ is strongly guarded and, by Lemma 4.3.12, λ is an external (quiescent) trace of $F[\tilde{E}/\tilde{Y}]$.

For the induction step suppose that ah is an external (quiescent) trace of $F[\tilde{E}^{n+1}[\tilde{P}/\tilde{X}]/\tilde{Y}]$ where $|h| = n$. From identity *2*, ah is an external (quiescent) trace of $F[\tilde{E}/\tilde{Y}][\tilde{E}^n[\tilde{P}/\tilde{X}]/\tilde{X}]$ and, by Lemma 4.3.14, $F[\tilde{E}/\tilde{Y}]$ is strongly guarded. From the definition of external trace and Lemmas 4.3.9 and 4.3.11 $\exists F_1, F_2$ such that

$$F[\tilde{E}/\tilde{Y}][\tilde{E}^n[\tilde{P}/\tilde{X}]/\tilde{X}] \xrightarrow{\tau^k} F_1[\tilde{E}^n[\tilde{P}/\tilde{X}]/\tilde{X}] \xrightarrow{a} F_2[\tilde{E}^n[\tilde{P}/\tilde{X}]/\tilde{X}]$$

where

$$F[\tilde{E}/\tilde{Y}] \xrightarrow{\tau^k} F_1[\tilde{X}] \xrightarrow{a} F_2[\tilde{X}]$$

and h is an external (quiescent) trace of $F_2[\tilde{E}^n[\tilde{P}/\tilde{X}]/\tilde{X}]$. By Lemma 4.3.14 and a simple induction argument F_2 is strongly compatible with $\tilde{E}$. By Lemmas 4.3.7 and 4.3.10

$$F[\tilde{E}/\tilde{Y}][\tilde{E}^n/\tilde{X}] \xrightarrow{\tau^k} F_1[\tilde{E}^n/\tilde{X}] \xrightarrow{a} F_2[\tilde{E}^n/\tilde{X}].$$

By induction h is an external (quiescent) trace of $F_2[\tilde{E}^n/\tilde{X}]$. Therefore, since by identity *2* $F[\tilde{E}^{n+1}/\tilde{Y}] = F[\tilde{E}/\tilde{Y}][\tilde{E}^n/\tilde{X}]$, ah is an external (quiescent) trace of $F[\tilde{E}^{n+1}/\tilde{Y}]$.

($\Leftarrow$) Suppose that λ is an external (quiescent) trace of $F[\tilde{E}/\tilde{Y}]$. By Lemma 4.3.14, $F[\tilde{E}/\tilde{Y}]$ is strongly guarded and, by Lemma 4.3.12, λ is an external (quiescent) trace of $F[\tilde{E}/\tilde{Y}][\tilde{P}/\tilde{X}]$. From identity *1*, λ is an external (quiescent) trace of $F[\tilde{E}[\tilde{P}/\tilde{X}]/\tilde{Y}]$.

For the induction step suppose that ah is an external (quiescent) trace of $F[\tilde{E}^{n+1}/\tilde{Y}]$ and suppose $|h| = n$. From identity *2*, ah is an external (quiescent) trace of $F[\tilde{E}/\tilde{Y}][\tilde{E}^n/\tilde{X}]$ and, by Lemma 4.3.14, $F[\tilde{E}/\tilde{Y}]$ is strongly guarded. From the definition of external trace and Lemmas 4.3.9 and 4.3.11, $\exists F_1, F_2$ such that

$$F[\tilde{E}/\tilde{Y}][\tilde{E}^n/\tilde{X}] \xrightarrow{\tau^k} F_1[\tilde{E}^n/\tilde{X}] \xrightarrow{a} F_2[\tilde{E}^n/\tilde{X}]$$

where

$$F[\tilde{E}/\tilde{Y}] \xrightarrow{\tau^k} F_1[\tilde{X}] \xrightarrow{a} F_2[\tilde{X}]$$

and h is an external (quiescent) trace of $F_2[\tilde{E}^n/\tilde{X}]$. By Lemma 4.3.14 and a simple induction argument F_2 is strongly compatible with $\tilde{E}$. By Lemmas 4.3.7 and 4.3.10

$$F[\tilde{E}/\tilde{Y}][\tilde{E}^n[\tilde{P}/\tilde{X}]/\tilde{X}] \xrightarrow{\tau^k} F_1[\tilde{E}^n[\tilde{P}/\tilde{X}]/\tilde{X}] \xrightarrow{a} F_2[\tilde{E}^n[\tilde{P}/\tilde{X}]/\tilde{X}].$$

By induction h is an external (quiescent) trace of $F_2[\tilde{E}^n[\tilde{P}/\tilde{X}]/\tilde{X}]$. Therefore, since by identity *2* $F[\tilde{E}^{n+1}[\tilde{P}/\tilde{X}]/\tilde{Y}] = F[\tilde{E}/\tilde{Y}][\tilde{E}^n[\tilde{P}/\tilde{X}]/\tilde{X}]$, ah is an external (quiescent) trace of $F[\tilde{E}^{n+1}[\tilde{P}/\tilde{X}]/\tilde{Y}]$.

■

We can finally prove Theorem 4.3.3.

Proof of Theorem 4.3.3 (recursive substitutivity)

1. Let h be an external (quiescent) trace of P_i and let $|h| = n$. By Lemma 4.3.5 part 2,

h is an external (quiescent) trace of $F[\tilde{E}^n[\tilde{P}/\tilde{X}]/\tilde{Y}]$ where $F \equiv Y_i$. By Lemma 4.3.15, h is an external (quiescent) trace of $F[E^n(\tilde{X})/\tilde{Y}]$ and, by Lemma 4.3.5 part 1, h is an external (quiescent) trace of $F[\tilde{X}/\tilde{Y}]$. Therefore h is an external (quiescent) trace of X_i and $Aut(X_i)$.

2. Let h be an external (quiescent) trace of $Aut(X_i)$, therefore an external (quiescent) trace of X_i, and let $|h| = n$. X_i can be expressed as $F[\tilde{X}/\tilde{Y}]$ where $F \equiv Y_i$. By Lemma 4.3.5 part 1, h is an external (quiescent) trace of $F[E^n(\tilde{X})/\tilde{Y}]$ and, by Lemma 4.3.15, h is an external (quiescent) trace of $F[\tilde{E}^n[\tilde{P}/\tilde{X}]/\tilde{Y}]$. Finally, by Lemma 4.3.5 part 3, h is an external (quiescent) trace of P_i.

∎

Chapter 5

An Axiomatization for the Quiescent Preorder

In this chapter we present the syntactic view of the theorems of Chapter 4 and we prove a completeness result for recursion-free expressions.

The first step consists in converting the theorems of Chapter 4 into actual axioms by giving syntactic approximations of the semantic auxiliary functions; then the completeness result can be stated and proved.

The completeness result is achieved through a special notion of normal form where the parallel operator is present. In general (see [ABV92]) the normal form contains only a 0 process, a prefixing operator and a nondeterministic choice operator. In DIOA the parallel operator cannot be eliminated in general from expressions of the form $\Omega \| nil$. The transition rules of DIOA, in fact, do not fit the format of [ABV92].

Once the normal form is identified, the completeness result is proven just for expressions in normal form and it is extended to general expressions by showing that each recursion-free expression with a finite interface has a provably equivalent one in normal form.

The rest of the chapter is organized as follows: Section 5.1 presents approximations for the auxiliary functions of Chapter 4 given in terms of the syntactic structure of the expressions. By substituting the new auxiliary functions in the theorems of Chapter 4 we obtain actual axioms; Section 5.2 presents some classes of expressions that are used for the completeness results; Section 5.3 presents other three axioms that can be easily stated using the notation of Section 5.2; Section 5.4 presents and proves the completeness result.

5.1 Syntactic definition of auxiliary functions

In this section we give an approximation of functions *Wsi*, *Wso*, *Localen*, *Quiet* and *Inten* that is based on the syntactic structure of an expression. The new functions we define can be substituted for the auxiliary functions used in Chapter 4 giving a set of actual axioms.

By looking at the way in which function *Wsi* is used in the theorems of Chapter 4, it is immediate to see that the approximation we need is an upper approximation of *Wsi*, i.e., we need a new function *wsi*, defined in terms of the syntactic structure of an expression e, such that, for every e, $Wsi(Aut(e)) \subseteq wsi(e)$. One specific property of *wsi* to guarantee the above relation is the following:

$$\text{if } a \in in(e) \text{ and } a \notin wsi(e) \text{ then } \exists e' \equiv_Q \Omega : e \stackrel{a}{\Longrightarrow} e'.$$

Table 5.1 contains the actual definition of *wsi* based on the property above. The definition of *wsi* is a bit complicated due to the presence of the two parameters A and B which are necessary for dealing with hiding and external choice operators. When dealing with the hiding operator it is not sufficient to look at the set *wsi* of its argument to establish the set *wsi* of the global expression: in fact all the hidden output actions must be considered internal. For this reason it is necessary to introduce an additional parameter A saying which actions should be considered internal in the evaluation of *wsi*. On the other hand, when dealing with an external choice context, not all traces with elements in A can be performed because some of them may be forbidden by the operator itself (for example e cannot perform the input action a in $e\ {}_{\emptyset}{+}_I\ f$). For the reason above it is necessary to introduce a second parameter B saying how the traces to consider should begin. Notice, however, that parameters A and B could be eliminated: the result is given by a coarser approximation of *Wsi* with the effect of a weaker set of axioms. The following lemma is characterizes the relationship between *Wsi* and *wsi*.

Lemma 5.1.1 *For each DIOA expression e, $Wsi(Aut(e)) \subseteq wsi(e)$.*

$$wsi_{A,B}(nil) = \emptyset$$

$$wsi_{A,B}(\Omega) = \emptyset$$

$$wsi_{A,B}(a\,.\,e) = \begin{cases} \{a\} & \text{if } a \in in(e)\backslash A \\ \emptyset & \text{if } a \in out(e) \cup A \end{cases}$$

$$wsi_{A,B}(e_1 \oplus e_2) = wsi_{A,B}(e_1) \cap wsi_{A,B}(e_2)$$

$$wsi_{A,B}(e_1\ {}_I{+}_J\ e_2) = \begin{cases} \emptyset & \text{if } B \cap A \cap (in(e_1)\backslash(I \cup J)) \neq \emptyset \\ (I \cap wsi_{A,B\cap(I\cup out(e_1))}(e_1)) \cup (J \cap wsi_{A,B\cap(J\cup out(e_2))}(e_2)) & \\ \quad \text{otherwise} & \end{cases}$$

$$wsi_{A,B}(\tau_I(e)) = wsi_{A\cup I,B}(e)$$

$$wsi_{A,B}(\rho(e)) = \rho(wsi_{\rho^{-1}(A),\rho^{-1}(B)}(e))$$

$$wsi_{A,B}(e_1 \| e_2) = wsi_{\emptyset,\emptyset}(e_1) \cup wsi_{\emptyset,\emptyset}(e_2)$$

$$wsi_{A,B}(X) = wsi_{A,B}(E(X))$$

Table 5.1: Definition of wsi for DIOA. $wsi(e) \stackrel{\text{def}}{=} wsi_{\emptyset,\emptyset}(e)$

Proof. The lemma is a direct consequence of the assertion

$$\text{if } a \in in(e) \text{ and } a \notin wsi(e) \text{ then } \exists e' \equiv_Q \Omega : e \stackrel{a}{\Longrightarrow} e'.$$

The assertion above is implied by the following one when choosing $A = \emptyset$:

$$\begin{aligned} &\text{if } a \in in(e)\backslash A \text{ and } a \notin wsi_{A,B}(e) \text{ and } B \subseteq ext(e) \\ &\text{then } \exists e' \equiv_Q \Omega \text{ and } h \in A^*, (h = \lambda \text{ or } first(h) \in B), \text{ and } e \stackrel{ha}{\Longrightarrow} e'. \end{aligned}$$

We show the last assertion by induction on the complexity of a guarded expression e. For unguarded expressions it is enough to substitute $E(X)$ for each unguarded occurrence of a process variable X.

The cases for nil and Ω are trivial since, for any input action, they both have only transitions to Ω. For the other operators we have the following cases:

Case 1 prefixing:

Let $e \equiv a\,.\,e'$ and suppose $b \notin wsi_{A,B}(e)$ where $b \in in(e)\backslash A$. By definition of wsi, $b \neq a$, hence the result is trivial since $a\,.\,e \xrightarrow{b} \Omega$ for any input action b different from a.

Case 2 internal choice:

Let $e \equiv e_1 \oplus e_2$ and suppose $a \notin wsi(e)$ where $a \in in(e)\backslash A$. By definition of wsi either $a \notin wsi(e_1)$ or $a \notin wsi(e_2)$. Suppose without loss of generality that $a \notin wsi(e_1)$. By induction there is $e_1' \equiv \Omega$ and $h \in A^*$ such that $h = \lambda$ or $first(h) \in B$, and $e_1 \stackrel{ha}{\Longrightarrow} e_1'$. By first using rule $\mathbf{ich}_1$ we have $e_1 \oplus e_2 \xrightarrow{\lambda} e_1 \stackrel{ha}{\Longrightarrow} e_1'$.

Case 3 external choice:

Let $e \equiv e_1\,{}_I{+}_J\,e_2$ and suppose $a \notin wsi_{A,B}(e)$ where $a \in in(e)\backslash A$. If $B \cap A \cap (in(e_1)\backslash(I \cup J)) \neq \emptyset$ then the result is trivial since $e_1\,{}_I{+}_J\,e_2 \xrightarrow{b} \Omega \xrightarrow{a} \Omega$ where $b \in B \cap A \cap (in(e_1)\backslash(I \cup J))$. If $B \cap A \cap (in(e_1)\backslash(I \cup J)) = \emptyset$ then one of the following cases holds:

1. $a \notin I \cup J$

   This case is trivial since $e_1\,{}_I{+}_J\,e_2 \xrightarrow{a} \Omega$.

2. $a \in I \cup J$ and $a \notin (J \cup wsi_{A,B\cap(I\cup out(e_1))}(e_1))$

   In this case we apply the induction hypothesis to e_1. Let e_1', h such that $e_1' \equiv \Omega$ and $e_1 \stackrel{ha}{\Longrightarrow} e_1'$. If $h = \lambda$ then rule $\mathbf{ech}_1$ can be used to derive $e_1\,{}_I{+}_J\,e_2 \stackrel{a}{\Longrightarrow} e_1'$ since $a \in I$; if $h \neq \lambda$ then, by induction, $first(h) \in I \cup out(e_1)$, hence rule $\mathbf{ech}_1$ can be used again.

3. $a \in I \cup J$ and $a \notin (I \cup wsi_{A,B\cap(J\cup out(e_2))}(e_2))$

   Similar to the previous case.

4. $a \in I \cup J$ and $a \notin wsi_{A,B\cap(I\cup out(e_1))}(e_1) \cup wsi_{A,B\cap(J\cup out(e_2))}(e_2)$

   In this case $a \in I$ or $a \in J$. Suppose without loss of generality that $a \in I$. The analysis is then the same as for item 2.

Case 4 hiding:

Let $e \equiv \tau_I(e')$ and let $a \notin wsi_{A,B}(e)$ where $a \in in(e)\backslash A$. By definition $wsi_{A,B}(\tau_I(e')) = wsi_{A\cup I,B}(e')$. By induction there exists $e'' \equiv_Q \Omega$ and $h' \in (A \cup I)^*$ such that $h' = \lambda$ or $first(h') \in B$, and $e' \stackrel{h'a}{\Longrightarrow} e''$. From the transition rules $\tau_I(e') \stackrel{ha}{\Longrightarrow} \tau_I(e'')$ where

$h = h' \lceil ext(e)$. Notice that, if $h' \neq \lambda$, then $first(h) \in B$ since $B \subseteq ext(e)$. In particular $\tau_I(e'') \equiv_Q \Omega$ and $h = \lambda$ or $first(h) \in B$.

Case 5 renaming:

Let $e \equiv \rho(e')$ and suppose $a \notin wsi_{A,B}(e)$ where $a \in in(e)\backslash A$. By definition $wsi_{A,B}(\rho(e')) = \rho(wsi_{\rho^{-1}(A),\rho^{-1}(B)}(e'))$, hence $\rho^{-1}(a) \notin wsi_{\rho^{-1}(A),\rho^{-1}(B)}(e')$ and $\rho^{-1}(a) \in in(e')\backslash\rho^{-1}(A)$. By induction there exists $e'' \equiv_Q \Omega$ and $h' \in \rho^{-1}(A)^*$ such that $h' = \lambda$ or $first(h') \in \rho^{-1}(B)$, and $e' \stackrel{h'\rho^{-1}(a)}{\Longrightarrow} e''$. From the transition rules $\rho(e') \stackrel{ha}{\Longrightarrow} \rho(e'')$ where $h = \rho(h')$. In particular $\rho(e'') \equiv_Q \Omega$ and $h = \lambda$ or $first(h) \in B$.

Case 6 parallel:

Let $e \equiv e_1 \| e_2$ and suppose $a \notin wsi_{A,B}(e)$ where $a \in in(e)\backslash A$. The conclusion follows directly by applying the induction hypothesis to both e_1 and e_2.

■

For function *Wso* we define an approximating function that satisfies the following property for each expression e:

$$\text{if } a \in out(e) \text{ and } \exists e' | e \stackrel{a}{\Longrightarrow} e' \text{ then } a \in wso(e).$$

Table 5.2 contains the actual definition of function *wso*. Unfortunately *wso* is not well defined for all DIOA expressions. Consider for example the process

$$X \stackrel{\text{def}}{=} \tau_{\{a\}}(a \,.\, (X \| nil))$$

where a is an output action of nil but not an action of X. The application of the definition of *wso* gives $wso(X) = wso(X)$. The problem is essentially due to the third case in the expression of $wso_{A,B}(a \,.\, e)$ where the prefix a is skipped and expression e is considered. One way to avoid the problem is to replace $wso_{A,A}(e)$ with $out(e)\backslash A$ in the expression for $wso_{A,B}(a \,.\, e)$; another way is to consider only those expressions for which *wso* is well defined, i.e., strongly guarded expressions as defined in Definition 4.3.1 of Chapter 4. On strongly guarded expressions the third case of the expression for $wso_{A,B}(a \,.\, e)$ does not cause any problem since a process variable will never be reached.

$$wso_{A,B}(nil) = \begin{cases} \emptyset & \text{if } A \cap B \cap in(nil) = \emptyset \\ out(nil)\backslash A & \text{otherwise} \end{cases}$$

$$wso(\Omega) = out(\Omega)\backslash A$$

$$wso_{A,B}(a\,.\,e) = \begin{cases} out(e)\backslash A & \text{if } B \cap A \cap \overline{\{a\}} \neq \emptyset \\ \{a\} \cap out(e) & \text{if } B \cap A \cap \overline{\{a\}} = \emptyset \text{ and } a \notin A \\ wso_{A,A}(e) & \text{if } B \cap A \cap \overline{\{a\}} = \emptyset \text{ and } a \in A \cap B \\ \emptyset & \text{if } B \cap A \cap \overline{\{a\}} = \emptyset \text{ and } a \in A\backslash B \end{cases}$$

$$wso_{A,B}(e_1 \oplus e_2) = wso_{A,B}(e_1) \cup wso_{A,B}(e_2)$$

$$wso_{A,B}(e_1 \;{}_I{+}_J\; e_2) = \begin{cases} wso_{A,B\cap(I\cup out(e_1))}(e_1) \cup wso_{A,B\cap(J\cup out(e_2))}(e_2) & \text{if } B \cap A \cap \overline{I \cup J} = \emptyset \\ out(e_1)\backslash A & \text{otherwise} \end{cases}$$

$$wso_{A,B}(\tau_I(e)) = wso_{A\cup I,B\cup I}(e)$$

$$wso_{A,B}(\rho(e)) = \rho(wso_{\rho^{-1}(A),\rho^{-1}(B)}(e))$$

$$wso_{A,B}(e_1 \| e_2) = \begin{cases} wso_{A,A}(e_1) \cup wso_{A,A}(e_2) & \text{if } \exists a \in B \cap A : a \in acts(e_1)\backslash ext(e_2) \\ & \quad \text{or } a \in acts(e_2)\backslash ext(e_1) \\ wso_{A,B}(e_1) \cup wso_{A,B}(e_2) & \text{otherwise} \end{cases}$$

$$wso_{A,B}(X) = wso_{A,B}(E(X))$$

Table 5.2: Definition of *wso* for DIOA $wso(e) \stackrel{\text{def}}{=} wso_{\emptyset,\emptyset}(e)$

The relationship between *Wso* and *wso* is then the following:

Lemma 5.1.2 *For every strongly guarded DIOA expression e, $Wso(Aut(e)) \subseteq wso(e)$.*

Proof. The lemma is a consequence of the assertion

$$\text{if } \exists e' : e \stackrel{a}{\Longrightarrow} e' \text{ for } a \in out(e), \text{ then } a \in wso(e).$$

The assertion above is implied by the following one when choosing $A = \emptyset$: if e is strongly guarded with respect to A and $\exists e', h$ such that $h \in A^*$, $h = \lambda$ or $first(h) \in B$, and $e \stackrel{ha}{\Longrightarrow} e'$ where $a \in out(e)\backslash A$, then $a \in wso_{A,B}(e)$. The lemma then follows by choosing $A = \emptyset$.

We show the last assertion by induction on the complexity of an expression e and we analyze each single operator. Clearly, since e is strongly guarded, e is not be a process variable.

Case 1 nil:

Let $e \equiv nil$ and suppose $\exists e', h \in A^*$ such that $h = \lambda$ or $first(h) \in B$, and $e \overset{ha}{\Longrightarrow} e'$ where $a \in out(e) \backslash A$. Since the only transitions for nil are labelled with input actions, it must be $h \neq \lambda$, $first(h) \in in(e)$ and $first(h) \in B$. This implies that $A \cap B \cap in(e) \neq \emptyset$. By definition, $wso_{A,B}(e) = out(e) \backslash A$, hence $a \in wso_{A,B}(e)$.

Case 2 omega:

This case is trivial since $wso_{A,B}(\Omega) = out(\Omega) \backslash A$.

Case 3 prefixing:

Let $e \equiv a \,.\, e'$ and suppose $\exists e'', h \in A^*$ such that $h = \lambda$ or $first(h) \in B$, and $e \overset{hb}{\Longrightarrow} e'$ where $b \in out(e) \backslash A$. We distinguish four cases:

1. $B \cap A \cap \overline{\{a\}} \neq \emptyset$

   This case is trivial since, by definition, $wso_{A,B}(e) = out(e) \backslash A$.

2. $B \cap A \cap \overline{\{a\}} = \emptyset$ and $a \notin A$

   In this case $h = \lambda$, hence a must be an output action and $b = a$. By definition $wso_{A,B}(e) = \{a\}$, hence $b \in wso_{A,B}(e)$.

3. $B \cap A \cap \overline{\{a\}} = \emptyset$ and $a \in A \cap B$

   In this case $h = ah'$ where $h' \in A^*$. In particular $a \,.\, e' \overset{a}{\longrightarrow} e'$, hence, by induction, $b \in wso_{A,A}(e')$. Notice, in fact, that e' is strongly guarded with respect to A. By definition $wso_{A,B}(e) = wso_{A,A}(e')$, hence $b \in wso_{A,B}(e)$.

4. $B \cap A \cap \overline{\{a\}} = \emptyset$ and $a \in A \backslash B$

   In this case $h = \lambda$. Moreover, since $a \in A$, b cannot exist.

Case 4 internal choice:

This case is a simple application of the induction hypothesis after observing that ha must be an external trace of one of the arguments of $\oplus$.

Case 5 external choice:

Let $e \equiv e_1 \,{}_I{+}_J\, e_2$ and suppose $e_1 \,{}_I{+}_J\, e_2 \overset{ha}{\Longrightarrow} e'$ where $h \in A^*$, $h = \lambda$ or $first(h) \in B$, and $a \in out(e) \backslash A$. We distinguish two cases:

1. $B \cap A \cap \overline{I \cup J} = \emptyset$

   In this case rule $\mathbf{ech}_3$ cannot be used for generating h, hence the only way to perform an output action is by first choosing between e_1 and e_2 using rules $\mathbf{ech}_{1,2}$. In particular the first external transition yielding ha is obtained by applying rule $\mathbf{ech}_1$ or $\mathbf{ech}_2$. Suppose without loss of generality that the applyed rule is $\mathbf{ech}_1$. In this case we have that $e_1 \stackrel{ha}{\Longrightarrow} e'$ and $h = \lambda$ or $first(h) \in I \cup out(e_1)$. By induction, then, $a \in wso_{A,B\cap(I\cup out(e_1))}(e_1)$. A symmetric argument holds if the applied rule is $\mathbf{ech}_2$.

2. $B \cap A \cap \overline{I \cup J} \neq \emptyset$

   This case is trivial since, by definition, $wso(e) = out(e)\backslash A$.

Case 6 hiding:

Let $e \equiv \tau_I(e')$ and suppose $\tau_I(e') \stackrel{ha}{\Longrightarrow} \tau_I(e'')$ where $h \in A^*$, $h = \lambda$ or $first(h) \in B$, and $a \in out(e)\backslash A$. By definition $\exists h' \in (A \cup I)^*$ such that $h' \lceil A = h$ and $e' \stackrel{h'a}{\Longrightarrow} e''$. Clearly, if $h' \neq \lambda$, $first(h') \in B \cup I$, hence, by induction, $a \in wso_{A\cup I, B\cup I}(e')$ giving $a \in wso_{A,B}(\tau_I(e'))$.

Case 7 renaming:

Let $e \equiv \rho(e')$ and suppose $\rho(e') \stackrel{ha}{\Longrightarrow} \rho(e'')$ where $h \in A^*$, $h = \lambda$ or $first(h) \in B$, and $a \in out(e)\backslash A$. By the transition rules $e' \stackrel{\rho^{-1}(ha)}{\Longrightarrow} e''$. Clearly, $\rho^{-1}(h) \in \rho^{-1}(A)^*$ and, if $\rho^{-1}(h) \neq \lambda$, $first(\rho^{-1}(h)) \in \rho^{-1}(B)$, hence, by induction, $\rho^{-1}(a) \in wso_{\rho^{-1}(A),\rho^{-1}(B)}(e')$ giving $a \in wso_{A,B}(\rho(e'))$.

Case 8 parallel:

Let $e \equiv e_1 \| e_2$. By definition

$$\underset{A,B}{wso}(e_1\|e_2) = \begin{cases} wso_{A,A}(e_1) \cup wso_{A,A}(e_2) & \text{if } \exists a \in B : a \in acts(e_1)\backslash ext(e_2) \\ & \quad \text{or } a \in acts(e_2)\backslash ext(e_1) \\ wso_{A,B}(e_1) \cup wso_{A,B}(e_2) & \text{otherwise} \end{cases}$$

Suppose $e_1\|e_2 \stackrel{ha}{\Longrightarrow} e'$ where $h \in A^*$, $h = \lambda$ or $first(h) \in B$, and $a \in out(e)\backslash A$. Suppose a is an output action of e_1 (the case for e_2 is analogous). By the transition rules it is a simple induction argument to see that, if e_1' is the left component of e', then $e_1 \stackrel{(h\lceil acts(e_1))a}{\Longrightarrow} e_1'$. If

$localen(nil) = \emptyset$
$localen(a.e) = \{a\} \cap out(e)$
$localen(e_1 \oplus e_2) = localen(e_1) \cup localen(e_2) \cup \{\tau\}$
$localen(e_1 \ {}_I{+}_J \ e_2) = localen(e_1) \cup localen(e_2)$
$localen(\tau_I(e)) = localen(e)$
$localen(\rho(e)) = \rho(localen(e))$
$localen(e_1 \Vert e_2) = localen(e_1) \cup localen(e_2)$
$localen(X) = localen(E(X))$
$inten(e) = true$ iff $\{\tau\} \in localen(e)$
$quiet(e) = true$ iff $localen(e) = \emptyset$

Table 5.3: Definition of *localen*, *inten* and *quiet*

$h = \lambda$ then, by induction, we immediately have that $a \in wso_{A,B}(e_1)$ and $a \in wso_{A,A}(e_1)$. If $first(h) \in acts(e_1)$ then again $a \in wso_{A,B}(e_1)$ and $a \in wso_{A,A}(e_1)$. If $first(h) \in acts(e_2) \backslash acts(e_1)$ then we can only conclude that $h \lceil acts(e_1) = \lambda$ or $first(h \lceil acts(e_1)) \in A$, hence $a \in wso_{A,A}(e_1)$. In all the cases the conclusion is that $a \in wso_{A,B}(e_1 \Vert e_2)$.

■

Remark 5.1.3 Functions *wsi* and *wso* could have been defined in several different ways. In this section we have just presented some arbitrary definition that, in our judgement, permit capturing the relationship between a large amount of expressions by means of the axioms of Section 4.2.

Functions *Localen*, *Inten* and *Quiet* can be easily defined in terms of the syntactic structure of an expression. Their definition is in table 5.3.

Lemma 5.1.4 *Given a DIOA expression e,*

1. $localen(e) = Localen(Aut(e))$,
2. $inten(e) = Inten(Aut(e))$ *and*
3. $quiet(e) = Quiet(Aut(e))$.

■

The following theorem is then straightforward.

Theorem 5.1.5 *The omega, renaming, prefixing, internal choice, external choice and hiding theorems for I/O automata are sound axioms for DIOA when expressions are interpreted as DIOA expressions and the syntactic auxiliary functions are substituted for the semantic auxiliary functions.* ■

5.2 Prefix forms

In this section we present some special classes of expressions called normal forms. The presentation also includes a definition of an unparameterized external choice operator which is useful for simplifying the notation.

Definition 5.2.1 (normal forms) A DIOA expression e is in *prefix normal form* if one of the following conditions holds.

1. $e \equiv \Omega \| nil \| \cdots \| nil$ (atomic expression)
2. $e \equiv a \,.\, e'$ where e' is in prefix normal form
3. $e \equiv e_1 \;{}_{wsi(e_1)}{+}_{wsi(e_2)}\; e_2$ where e_1 and e_2 are in prefix normal form but not atomic.

A DIOA expression e is in *internal prefix form* if $e \equiv e_1 \oplus \cdots \oplus e_n$ where each e_i is in prefix normal form. We abbreviate $e_1 \oplus \cdots \oplus e_n$ with $\bigoplus e_i$. ■

The reason for the complexity of item 1 is that in general the parallel operator cannot be eliminated from an atomic expression.

When dealing with expressions in prefix normal form it is possible to drop the parameters from the external choice operator; moreover, when e is not an atomic expression different from nil, it is possible to use the notation $e \equiv \sum_{i \in I} a_i \,.\, e_i$ where $I = \emptyset$ means $e \equiv nil$.

The above idea also suggests the use of an unparameterized choice operator $+$ to simplify the notation for expressions when possible: $e + f$ is defined to be $e \;{}_{wsi(e)}{+}_{wsi(f)}\; f$.

5.3 Other axioms

In this section we present other three important axioms which can be easily stated using the prefix normal form. The first two axioms are the expansion axioms, giving the possibility to convert a parallel composition of n expressions into a nondeterministic composition of expressions.

Proposition 5.3.1 (expansion axioms) *The following axioms are sound:*

$\mathbf{E}_1$ *Let $e \equiv \Omega_{S_0} \| nil_{S_1} \| \cdots \| nil_{S_n}$ be of sort S. For each $a \in out(S_0) \cup in(S)$ let e_a be the unique state that e reaches with action a. Then $e \equiv_Q (\sum_{a \in out(S_0) \cup in(S)} a \,.\, e_a) \oplus (\sum_{a \in in(S)} a \,.\, e_a)$.*

$\mathbf{E}_2$ *Let $e \equiv e_1 \| e_2 \| \cdots \| e_n$ where each e_i is of the form $\sum_j a_{ij} \,.\, e_{ij}$. For each action $a \in ext(e)$ let*

$$E_a^i = \begin{cases} \{e_{ij} | a_{ij} = a\} & \textit{if } a \in acts(e_i) \\ \{e_i\} & \textit{otherwise} \end{cases}$$

Let $out(a)$ be the index j such that a is an output action of j (0 otherwise) and let

$$E_a = \begin{cases} \emptyset & \textit{if } out(a) \neq 0 \textit{ and } E_a^{out(a)} = \emptyset \\ \{f_1 \| \cdots \| f_n : f_i \in E_a^i \vee (E_a^i = \emptyset \wedge f_i \equiv \Omega)\} & \textit{otherwise} \end{cases}$$

Then $e \equiv_Q \sum_{a \in ext(e)} (\sum_{f \in E_a} a.f)$.

■

The third axiom concerns atomic expressions. We also prove that the axiom below completely characterizes the quiescent preorder for internal choice compositions of atomic expressions.

Proposition 5.3.2 (completeness axiom) *The following assertion is valid:*

$\mathbf{Cp}_1$ *Let $e_i, 0 \leq i \leq n$ be atomic expressions and, for each action a, let f_i^a be the state that e_i reaches with action a ($\bullet$ if no state exists). Then $e_0 \sqsubseteq_Q \sum_{1 \leq i \leq n} e_i$ iff, for each action a, either*

1. *$f_i^a \equiv e_i, 0 \leq i \leq n$ or*
2. *$f_0^a \equiv \bullet$ or*
3. *$f_0^a \sqsubseteq_Q \sum_{f_i^a \not\equiv \bullet} f_i^a$.*

Proof.

Soundness

Suppose, for each action a, one of the conditions 1, 2 or 3 to be valid. Let t be an external (quiescent) trace of e_0. The case for $t = \lambda$ is trivial since λ is a quiescent trace of any atomic expression. Let $t = t_1 t_2$ where t_1 is the longest prefix of t such that each $e_i \xrightarrow{t_1} e_i$ by means of self loop transitions. If $t_2 = \lambda$ then trivially t is an external (quiescent) trace of $(\sum_{1 \leq i \leq n} e_i)$ using the same argument as for λ. Suppose $t_2 = at_3$ for some action a and let $e_0 \xrightarrow{a} f_0^a$. t_3 is then an external (quiescent) trace of f_0^a and, by hypothesis and the definition of t_2, t_3 is an external (quiescent) trace of $(\sum_{f_i^a \not\equiv \bullet} f_i^a)$ and $\{f_i^a \not\equiv \bullet\} \neq \emptyset$ (in fact conditions 1 and 2 are false). This implies that $\exists j : t_3$ is an external (quiescent) trace of f_j^a. Moreover $(\sum_{1 \leq i \leq n} e_i) \stackrel{\lambda}{\Longrightarrow} e_j \xrightarrow{t_1} e_j \xrightarrow{a} f_j^a$, hence t is an external (quiescent) trace of $(\sum_{1 \leq i \leq n} e_i)$.

Completeness

Let $e_0 \sqsubseteq_Q (\sum_{1 \leq i \leq n} e_i)$ and suppose conditions 1, 2 and 3 to be false for some action a. Since, by condition 2, $f_0^a \not\equiv \bullet$, we have that $e_0 \xrightarrow{a} f_0^a$. Since condition 3 is false, then either $\{f_i^a \not\equiv \bullet\} = \emptyset$ or $f_0^a \not\sqsubseteq_Q (\sum_{f_i^a \not\equiv \bullet} f_i^a)$. The first case cannot hold, for which otherwise a is an external trace of e_0 but not an external trace of $(\sum_{1 \leq i \leq n} e_i)$. Let $t = at'$ where t' is an external (quiescent) trace of f_0^a but not an external (quiescent) trace of $(\sum_{f_i^a \not\equiv \bullet} f_i^a)$. We show that t is not an external (quiescent) trace of $(\sum_{1 \leq i \leq n} e_i)$. Suppose the contrary. By Lemma 5.4.3, t is an external (quiescent) trace of e_i for some $i > 0$. In particular $e_i \xrightarrow{a} f_i^a$, hence t' is an external (quiescent) trace of f_i^a, i.e., t' is an external (quiescent) trace of $\sum_{f_i^a \not\equiv \bullet} f_i^a$, absurdum. ∎

5.4 Completeness results

In this section we prove the completeness result for recursion-free expressions. It is achieved through the following steps:

1. the completeness result is shown for expressions in internal prefix form.

2. each recursion-free expression is shown to have a provably equivalent expression in internal prefix form;

The main theorem is then the following:

Theorem 5.4.1 (completeness) *Let e, f be recursion-free DIOA expressions with a finite interface. If $e \sqsubseteq_Q f$ then $\mathcal{A} \vdash e \sqsubseteq_Q f$ where $\mathcal{A}$ is the set of all axioms presented in this thesis.* ■

The completeness result for expressions in internal prefix form is shown through an additional axiom. We prove its soundness by using the axiom version of the theorems of Chapter 4. We first state some simple lemmas.

Lemma 5.4.2 *Let $e \equiv \sum_{i \in I} a_i \,.\, e_i$. Then*

$$wsi(e) = \{a_i : i \in I\} \cap in(e) \text{ and}$$
$$wso(e) = \{a_i : i \in I\} \cap out(e).$$

Proof. Direct application of the definitions of *wsi* and *wso*. ■

Lemma 5.4.3 *Let $e \equiv \bigoplus_{i \in I} e_i$. Then*

1. *$etraces(e) = \cup_{i \in I}\, etraces(e_i)$ and*

2. *$qtraces(e) = \cup_{i \in I}\, qtraces(e_i)$.*

Proof. Simple consequence of the transition rules for $\oplus$. ■

Proposition 5.4.4 (completeness axiom) *The following assertion is valid:*

Cp$_2$ *Let $e \equiv \sum_i a_i \,.\, e_i$ and $f \equiv \bigoplus_j f_j$ where $f_j \equiv \sum_k b_{j\,k} \,.\, f_{j\,k}$. For each a, j let*

$$g_j^a \equiv \begin{cases} \bigoplus_{b_{j\,k}=a} f_{j\,k} & \text{if } \{k | b_{j\,k} = a\} \neq \emptyset \\ \bullet & \text{otherwise} \end{cases}$$

Then $e \sqsubseteq_Q f$ iff the following three conditions hold:

(a) $quiescent(e) \Longrightarrow \exists j : quiescent(f_j)$

(b) $\forall i \left(e_i \sqsubseteq_Q \bigoplus_{g_j^{a_i} \not\equiv \bullet} g_j^{a_i} \text{ and } \exists j : g_j^{a_i} \not\equiv \bullet\right)$ or $(a_i \in in(e) \text{ and } \exists j : g_j^{a_i} \equiv \bullet)$

(c) $\forall a \in \bigcap(wsi(f_j)) \setminus wsi(e) \quad \Omega \sqsubseteq_Q \bigoplus_j g_j^a$

Proof.

Soundness

Suppose conditions 1, 2 and 3 to be valid. We perform the following quiescent equivalence preserving transformations on e and f:

1. Using axiom $\mathbf{Ec}_9$ add $a\,.\,\Omega$ to each expression f_j such that $a \notin wsi(f_j)$ and $a \in wsi(e) \cup wsi(f)$. Do the same on e.

2. Using axiom $\mathbf{Ec}_{13}$ replicate on all the f_js each summand $a\,.\,f'_k$ of each f_k where a is an input action. For example $(a\,.\,f'_1 + f''_1) \oplus f_2 \oplus \cdots \oplus f_n$ becomes $(a\,.\,f'_1 + f''_1) \oplus (a\,.\,f'_1 + f_2) \oplus \cdots \oplus (a\,.\,f'_1 + f_n)$

3. Repeat the operation of 2 for summands $a\,.\,f'_k$ where a is an output action. Only non quiescent expressions can be considered.

4. Using axiom $\mathbf{Ec}_{13}$ group all expressions with a common prefix in each expression f_j.

5. Reduce to $a\,.\,\Omega$ each summand of the form $a\,.\,(\Omega \oplus \cdots)$ of each f_j. This step is possible since it is immediate to prove $e \equiv_Q e \oplus \Omega$ by using axioms $\mathbf{M}$ and $\mathbf{Ic}_8$.

6. Merge equal expressions on the f-side using axiom $\mathbf{Ic}_3$.

The new expressions $e' \equiv_Q e$ and $f' \equiv_Q f$ coming out from the above manipulations are

$$e' \equiv e + \sum_{a \in wsi(f) \setminus wsi(e)} a\,.\,\Omega$$

and

$$f' \equiv (f'' + \sum_{a \in A} a\,.\,f''_a) \oplus f''$$

where A is a set of output actions,

$$f'' \equiv (\sum_{a \in wsi(e) \cup wsi(f)} a\,.\,f''_a),$$

and each f''_a is

$$f''_a \equiv \begin{cases} \sum_{g^a_j \not\equiv \bullet} g^a_j & \text{if } a \in out(e) \text{ or } (a \in in(e) \text{ and } \nexists j | g^a_j \equiv \bullet) \\ \Omega & \text{if } a \in in(e) \text{ and } \exists j | g^a_j \equiv \bullet) \end{cases}$$

Notice that the right expression f' appears only if there is at least a quiescent f_j. We now distinguish two cases:

1. e is quiescent

   In this case e' is also quiescent and, by hypothesis there is a quiescent f_j. We prove that $e' \sqsubseteq_Q f'$. Axiom $\mathbf{Ic}_8$ is then sufficient to conclude. We show in particular that, for each summand $a \,.\, e''$ of e', $e'' \sqsubseteq_Q f'_a$. Axiom $\mathbf{Ec}_3$ and substitutivity are then sufficient to conclude. If $\exists j | g^a_j \equiv \bullet$ then $f'_a \equiv \Omega$ and axiom $\mathbf{M}$ is sufficient to conclude; if otherwise, then $f'_a \equiv \sum_{g^a_j \not\equiv \bullet} g^a_j$. If $a.e''$ is a summand of e then the conclusion follows from hypothesis; if otherwise then the conclusion follows from hypothesis again after observing that $a \in \bigcap(wsi(f_j)) \setminus wsi(e)$.

2. e is not quiescent

   In this case we prove that $e' \sqsubseteq_Q f' + \sum_{a \in A} a \,.\, f'_a$. The method is exactly the same we used in the first case. For any summand $a \,.\, e''$ of e', in fact, there is a summand $f \,.\, f'_a$ of $f' + \sum_{a \in A} a \,.\, f'_a$. Additional summands $a \,.\, f'_a$ of the right expression that do not have any correspondent summand in e' can be added using axiom $\mathbf{Ec}_5$.

Completeness

Let $e \sqsubseteq_Q f$. We show that conditions 1,2 and 3 are satisfied.

1. Suppose e to be quiescent. By definition of quiescent trace, λ is a quiescent trace of e, hence, by hypothesis, λ is a quiescent trace of f. By Lemma 5.4.3, λ is a quiescent trace of f_j for some j, hence, since f_j does not enable any internal action, f_j is quiescent.

2. Suppose condition 2 to be false and let i be one of the indexes for which the condition is false. We distinguish the following cases:

   (a) a_i is an output action

In this case the left side of condition 2 must be false. If $\forall j : g_j^{a_i} \equiv \bullet$, then no external trace with a_i as first action is an external trace for f, while a_i is an external trace of e. This gives a contradiction, hence $\exists j : g_j^{a_i} \not\equiv \bullet$. Since condition 2 is false, it must be $e_i \not\sqsubseteq_Q (\sum_{g_j^{a_i} \not\equiv \bullet} g_j^{a_i})$. Let t' be an external (quiescent) trace of e_i but not of $\sum_{g_j^{a_i} \not\equiv \bullet} g_j^{a_i}$. Clearly $t = a_i t'$ is an external (quiescent) trace of e. We show that t is not an external (quiescent) trace of f obtaining a contradiction. Suppose $f \stackrel{a}{\Longrightarrow} f'$ where t' is an external (quiescent) trace of f'. From the transition rules, $\exists j, k : f' \equiv f_{j\,k}$ and $a_{j\,k} = a_i$. By definition, $f_{j\,k}$ is a summand of $g_j^{a_i}$, hence t' is an external (quiescent) trace of $\sum_{g_j^{a_i} \not\equiv \bullet} g_j^{a_i}$. This gives a contradiction.

(b) a_i is an input action

Since the right part of condition 2 must be false, then $\forall j : g_j^{a_i} \not\equiv \bullet$. It is then enough to repeat the argument of the previous case to conclude.

3. Suppose condition 3 to be false. Then $\exists a \in \bigcap(wsi(f_j)) \setminus wsi(e) : \Omega \not\sqsubseteq_Q (\sum_j g_j^a)$. Let t' be an external (quiescent) trace of Ω but not of $\sum_j g_j^a$, and consider $t = at'$. Clearly, since from the transition rules and Lemma 5.4.2 $e \stackrel{a}{\longrightarrow} \Omega$, t is an external (quiescent) trace of e. By using the same argument as in case (b) of the proof for condition 2 we obtain that t is an external (quiescent) trace of $\sum_j g_j^a$. This gives a contradiction.

■

The following definition is fundamental for setting up the opportune inductive proofs.

Definition 5.4.5 (complexities) The *atomic complexity* $\mathcal{A}$ of an atomic expression e is the number of *nil* subexpressions appearing in e.

The *prefix complexity* $\mathcal{P}$ of an expression e in prefix normal form is defined as

$$\mathcal{P}(e) = \begin{cases} 0 & \text{if } e \text{ is atomic} \\ 1 + \mathcal{P}(e_1) & \text{if } e \equiv a \,.\, e_1 \text{ for some action } a \\ max(\mathcal{P}(e_1), \mathcal{P}(e_2)) & \text{if } e \equiv e_1 + e_2 \end{cases}$$

The *complexity* $\mathcal{C}$ of an expression e in internal prefix form is the maximum prefix complexity of its summands. ■

We first prove the completeness result for atomic expressions.

Lemma 5.4.6 *Let e be an atomic expression. If $e \xrightarrow{a} f$ for some external action a where $e \not\equiv f$, then there is an atomic expression f' such that $\mathcal{A}(f) < \mathcal{A}(e)$ and $\vdash f \equiv_Q f'$.*

Proof. From the transition rules a process Ω only has self loops for external actions. If $e \not\equiv f$, then the only processes that can have changed are *nil*. A process *nil* can either have a self loop or a transition to Ω. This implies that at least one of the *nil* subterms of e has became Ω in f. From axiom **P** all Ω subexpressions of f can be collapsed into a single Ω expression. The resulting expression (f') is atomic and is such that $\mathcal{A}(f) < \mathcal{A}(e)$. ∎

Lemma 5.4.7

$$e_1 \oplus \cdots \oplus e_n \sqsubseteq_Q f \textit{ iff } \forall_{1 \leq i \leq n} e_i \sqsubseteq_Q f.$$

Proof. Direct consequence of Lemma 5.4.3. ∎

Lemma 5.4.8 (completeness for atomic expressions) *Let e, f be internal sums of atomic expressions. If $e \sqsubseteq_Q f$ then $\vdash e \sqsubseteq_Q f$.*

Proof. From Lemma 5.4.7 and axiom $\mathbf{Ic}_3$ it is sufficient to analyze the case in which e is atomic. We show the result by induction on the sum n of the atomic complexities of e and the summands of f. If $n = 0$ then $e = \Omega$ and each summand of f is Ω. By axiom $\mathbf{Ic}_3$, $\vdash f \equiv_Q \Omega$, hence, by reflexivity and transitivity of $\sqsubseteq_Q$, $\vdash e \sqsubseteq_Q f$. Let $n > 0$. Since $e \sqsubseteq_Q f$, by Lemma 5.3.2 the premises of axiom $\mathbf{Cp}_1$ are satisfied. For each action a condition 1 and 2 are easily checkable. Suppose conditions 1 and 2 to be false. Then condition 3 is true. By Lemma 5.4.6 and the non validity of condition 1, the sum of the atomic complexities of the expressions to compare on condition 3 is less than n. It is then enough to apply the induction hypothesis and use axiom $\mathbf{Cp}_1$ to conclude. ∎

We can now prove the completeness result for expressions in prefix normal form.

Proposition 5.4.9 (completeness for expressions in internal prefix form) *Let e and f be expressions in internal prefix form. If $e \sqsubseteq_Q f$ then $\vdash e \sqsubseteq_Q f$.*

Proof. From Lemma 5.4.7 and axiom $\mathbf{Ic}_3$ it is sufficient to analyze the case in which e is in prefix normal form. We show the result by induction on the maximum complexity n of e and f. If $n = 0$ then e and the summands of f are atomic expressions and the result is given by Lemma 5.4.8. If $n > 0$ then, by using axiom $\mathbf{E}_1$, there are two expressions e', f' such that $\vdash e \equiv_Q e'$, $\vdash f \equiv_Q f'$, the maximum complexity of e' and f' is n, and no summands of e' and f' are atomic expressions. We can again assume e' to be in prefix normal form. By applying axiom $\mathbf{Cp}_2$ to e' and f' we have that, for each condition involving the comparison of some expressions, one level of prefixing is eliminated, hence the complexity of the expressions to prove in relation is less than n. By applying the induction hypothesis and successively axiom $\mathbf{Cp}_2$, the proof is concluded. ■

To prove that every recursion-free expression has a provably equivalent one in internal prefix form we show that the class of expressions in internal prefix form is closed under all the operators of DIOA.

Lemma 5.4.10 (closure under internal choice) *The internal prefix form is closed under internal choice.*

Proof. Immediate from the definition of internal prefix form and the associativity of the internal choice operator. ■

Lemma 5.4.11 (closure under prefixing) *Let e be an expression in internal prefix form. Then there is an expression g in internal prefix form such that $\vdash a\,.\,e \equiv_Q g$.*

Proof. Direct consequence of the distributivity of $a.$ over $\oplus$ (axiom $\mathbf{Ic}_4$). ■

Lemma 5.4.12 (closure under external choice) *Let e, f be expressions in internal prefix form. Then there is an expression g in internal prefix form such that $\vdash e\ {}_I{+}_J\ f \equiv_Q g$.*

Proof. By repeatedly using axiom $\mathbf{Ic}_5$ (distributivity of ${}_I{+}_J$ over $\oplus$) the problem is reduced to the case in which e and f are in prefix normal form. If e or f are atomic expressions, then we use axiom $\mathbf{E}_1$ to transform them into non atomic expressions e', f' in prefix normal form. By means of axiom $\mathbf{Ec}_{14}$ the operator ${}_I{+}_J$ is replaced by ${}_K{+}_K$ where $K = wsi(e') \cap wsi(f')$. By repeatedly applying axiom $\mathbf{Ec}_{16}$ (and axiom $\mathbf{Ec}_2$) we obtain $\vdash e'\ {}_K{+}_K\ f' \equiv_Q e''\ {}_K{+}_K\ f''$ where one of the following conditions hold:

1. $wsi(e'') = wsi(f'') = K$

   In this case we already have our expression g.

2. $wsi(e'') = K$, $f'' \equiv a \,.\, f'''$, a is an input action and $a \notin K$

   In this case axiom $\mathbf{Ec}_{15}$ is sufficient to conclude.

3. $wsi(f'') = K$, $e'' \equiv a \,.\, e'''$, a is an input action and $a \notin K$

   In this case axioms $\mathbf{Ec}_{2,15}$ are sufficient to conclude.

4. $e'' \equiv a \,.\, e'''$, $f'' \equiv b \,.\, f'''$, a, b are input actions and $a, b \notin K$

   In this case $K = \emptyset$, hence we use axioms $\mathbf{Ec}_{2,15,16}$ to show the following:

$$e'' \,{}_{\emptyset}{+}_{\emptyset}\, f'' \equiv_Q (e'' \,{}_{\emptyset}{+}_{\emptyset}\, nil) \,{}_{\emptyset}{+}_{\emptyset}\, f'' \equiv_Q (nil \,{}_{\emptyset}{+}_{\emptyset}\, e'') \,{}_{\emptyset}{+}_{\emptyset}\, f'' \equiv_Q nil \,{}_{\emptyset}{+}_{\emptyset}\, f'' \equiv_Q nil.$$

   The assertion on the complexity is then trivial.

This concludes the proof. ∎

Lemma 5.4.13 (closure under hiding) *Let e be an expression in internal prefix form. Then there is an expression g in internal prefix form such that $\vdash \tau_I(e) \equiv_Q g$.*

Proof. By repeatedly using axiom $\mathbf{Ic}_6$ (distributivity of τ_I over $\oplus$) the problem is reduced to the case in which e is in prefix normal form. The proof is by induction on the prefix complexity of e. If e is atomic then, by repeatedly using axiom $\mathbf{I}_{14}$ and the substitutivity property, we obtain an expression e' such that $\vdash \tau_I(e) \equiv_Q \tau_I(e')$ and $\tau_I(e')$ satisfies the conditions for axiom $\mathbf{I}_{15}$. The application of axiom $\mathbf{I}_{15}$ yields the desired expression g. Notice that the complexity of g is 0. Suppose now the prefix complexity of e to be $n > 0$, i.e. $e \equiv (\sum_J a_j \,.\, e_j)$ where the prefix complexity of each e_j is less than n. We distinguish the following cases:

1. $\forall_j a_j \notin I$

   By using axioms $\mathbf{I}_{3,4}$ we have $\vdash \tau_I(\sum_J a_j \,.\, e_j) \equiv_Q (\sum_J a_j \,.\, \tau_I(e_j))$. By induction each $\tau_I(e_j)$ has a provably equivalent expression g_j in internal prefix form. By Lemma 5.4.11 each $a_j \,.\, g_j$ has a provably equivalent expression g'_j in internal prefix form. The desired expression g is then $(\sum_J g'_j)$. The condition on the complexity is trivially satisfied.

2. $e \equiv e' + a_n \, . \, e_n$ where e' is quiescent and $a_n \in I$

   From axiom $\mathbf{I}_{13}$, $\vdash \tau_I(e) \equiv_Q \tau_I(e' \;{}_{wsi(e')}{+}_{wsi(e')}\; e_n)$. From case 1, $\vdash \tau_I(e') = e''$ for some e'' in internal prefix form. By induction $\vdash \tau_I(e_n) \equiv_Q e'_n$ for some e'_n in internal prefix form. By using axiom $\mathbf{E}_1$ we can force e'' and e'_n not to have atomic summands. From axioms $\mathbf{I}_{15,3,4}$ and $\mathbf{Ic}_5$ there are two expressions e''' and e''_n, differing only in the signatures of the operators, such that $\vdash e'' \equiv_Q \tau_I(e''')$ and $\vdash e'_n \equiv_Q \tau_I(e''_n)$. In particular e''' and e''_n do not enable actions from I. From axioms $\mathbf{I}_{9,4}$ $\vdash \tau_I(e' \;{}_{wsi(e')}{+}_{wsi(e')}\; e_n) \equiv_Q \tau_I(e''' \;{}_{wsi(e')}{+}_{wsi(e')}\; e''_n) \equiv_Q \tau_I(e''') \;{}_{wsi(e')}{+}_{wsi(e')}\; \tau_I(e''_n) \equiv_Q e'' \;{}_{wsi(e')}{+}_{wsi(e')}\; e'_n$. The closure under external choice is then sufficient to conclude.

3. $e \equiv a_1 \, . \, e_1$ where $a_1 \in I$

   By induction $\vdash \tau_I(e_1) \equiv_Q e'_1$ for some e'_1 in internal prefix form. By using axiom $\mathbf{E}_1$ we can force e'_1 not to have atomic summands. Moreover, from the internal choice axioms, we can assume without loss of generality that e'_1 is in prefix normal form. From axioms $\mathbf{I}_{15,3,4}$ and $\mathbf{Ic}_5$ there is an expressions e''_1, differing only in the signatures of the operators, such that $\vdash e'_1 \equiv_Q \tau_I(e''_1)$. In particular e''_1 does not enable actions from I. From axiom $\mathbf{I}_8$, $\vdash \tau_I(a_1 \, . \, e_1) \equiv_Q \tau_I(a_1 \, . \, e''_1)$. From axiom $\mathbf{Ec}_{15}$, $\vdash a_1 \, . \, e''_1 \equiv_Q nil + a_1 \, . \, e''_1$. From axiom $\mathbf{I}_{13}$, $\vdash \tau_I(nil + a_1 \, . \, e''_1) = \tau_I(nil \;{}_{\emptyset}{+}_{\emptyset}\; e''_1)$. By using axiom $\mathbf{Ec}_{16}$ all input prefixed summands of e''_1 can be eliminated obtaining $\vdash \tau_I(nil \;{}_{\emptyset}{+}_{\emptyset}\; e''_1) \equiv_Q \tau_I(nil \;{}_{\emptyset}{+}_{\emptyset}\; e'''_1)$ where $wsi(e'''_1) = \emptyset$. From axiom $\mathbf{Ec}_5$ $\vdash \tau_I(nil \;{}_{\emptyset}{+}_{\emptyset}\; e'''_1) \equiv_Q \tau_I(e'''_1)$. The application of axioms $\mathbf{I}_{15,3,4}$ is then sufficient to conclude.

4. $e \equiv e' + a_n \, . \, e_n$ where e' is not quiescent and $a_n \in I$

   From axioms $\mathbf{I}_{12}$ and $\mathbf{Ic}_6$, $\vdash \tau_I(e) \equiv_Q \tau_I(e' \oplus e_n) \equiv_Q \tau_I(e') \oplus \tau_I(e_n)$. The expression $\tau_I(e_n)$ can be reduced by induction. For the expression $\tau_I(e')$ we observe that e' has one summand less than e. We then repeatedly apply case 4 to $\tau_I(e')$ and to its derived expressions until case 4 does not apply (and we know that case 4 will not apply at a certain point since at least two summands are needed). When case 4 does not apply, we use the applicable case between 1,2 and 3 and the proof is concluded.

■

Lemma 5.4.14 (closure under renaming) *Let e be an expression in internal prefix form. Then there is an expression f in internal prefix form such that* $\vdash \rho(e) \equiv_Q f$.

Proof. Since the renaming operator is distributive over all other DIOA operators, it can be pushed down to the lowest level and then be completely eliminated from any DIOA expression. ∎

Lemma 5.4.15 (closure under parallel composition) *Let e, f be expressions in internal prefix form with a finite interface. Then there is an expression g in internal prefix form such that* $\vdash e \| f \equiv_Q g$.

Proof. By repeatedly using axiom $\mathbf{Ic}_7$ (distributivity of $\|$ over $\oplus$) the problem is reduced to the case in which e and f are in prefix normal form. We proceed by induction on the prefix complexities of e and f. If both e and f are atomic then the result is immediate. Suppose now the maximum complexity of e and f to be $n > 0$. If e or f are atomic expressions, then we use axiom $\mathbf{E}_1$ to transform them into expressions e', f' in internal prefix form that have no atomic summands without affecting the maximum complexity of e and f. After reducing again the problem to the case in which all expressions are in prefix normal form, we apply the expansion axiom $\mathbf{E}_2$ obtaining a new equivalent expression $e' \equiv \sum_{j \in J} a_j \,.\, f_j$ where each $f_j \equiv f_j^1 \| f_j^2$ and the maximum complexity of f_j^1 and f_j^2 is less than n. It is then enough to apply the induction hypothesis and use axioms $\mathbf{Ic}_{4,5}$ to conclude. ∎

Lemma 5.4.16 (reduction to internal prefix form) *Let e be a recursion-free DIOA expression with a finite interface. Then there is an expression g in internal prefix form such that* $\vdash e \equiv_Q g$.

Proof. The proof proceeds by structural induction of the given expression e. The basic cases nil and Ω are trivial since they are atomic expressions. For all other operators we first reduce their arguments using the induction hypothesis, then we eliminate the new operator by means of the closure lemmas 5.4.10, 5.4.11, 5.4.12, 5.4.13, 5.4.14 and 5.4.15. ∎

We can finally prove the main theorem.

Theorem 5.4.17 (completeness) *Let e, f be recursion-free DIOA expressions with a finite interface. If $e \sqsubseteq_Q f$ then $\mathcal{A} \vdash e \sqsubseteq_Q f$ where $\mathcal{A}$ is the set of all axioms presented in this thesis.*

Proof. By means of Lemma 5.4.16 the problem is reduced to the case in which e and f are in internal prefix form. The completeness result is then stated by Proposition 5.4.9. ∎

Chapter 6

Example Specifications and Verifications

In this chapter we show some example specifications and verifications within DIOA. We specify a simple circuit that is reported in [Jos92] and a more complicated one that is reported in [BV88]. The examples are preceded by a discussion about the use of the quiescent preorder as an implementation relation.

6.1 Quiescent preorder as an implementation relation

The intuitive idea of implementation at the base of the semantics of I/O automata is that an implementation must respond to a sequence of external stimuli with some output actions whenever the specification must too. The way in which the above idea is captured is by means of fair trace inclusion.

Can the quiescent preorder be used for capturing the same idea of implementation? In this section we just want to give an informal understanding of this question without pretending to be formal. With this discussion we want to point out some of the problems of chosing a relation as an implementation relation.

The answer to the given question is "no" in general. The absence of the notion of fairness, in fact, causes several problems. Consider for example

$$A \stackrel{\text{def}}{=} \tau_{\{i\}}(a \,.\, X)$$

and

$$B \stackrel{\text{def}}{=} a \,.\, b \,.\, nil$$

where $X \stackrel{\text{def}}{=} i \,.\, X$, a is an input action and b is an output action. It is immediate to verify that $A \sqsubseteq_Q B$, but we do not want to consider A to be an implementation of B since A refuses to perform action b after receiving the input a while B must perform the output action b. The problem is essentially in the internal looping of A since we cannot observe it by means of external and quiescent traces. In I/O automata the distinction between A and B is given by fair traces: in fact a is a fair trace of A but not a fair trace of B according to the I/O automata semantics. Also in receptive process theory [Jos92] the problem is solved since a is a divergence of A but not a divergence of B. The use of divergences, however, leads to $A \not\sqsubseteq B + a \,.\, nil$ while the quiescent and fair preorders lead to $A \sqsubseteq B + a . nil$. We would like to consider $A \sqsubseteq B + a . nil$ since, although the implementation A refuses to perform action b after a, the specification may too.

In order to use the quiescent preorder we have to be sure that situations like the one presented above do not arise, i.e., we can deal only with processes that, whenever they present an internal divergence, they can reach a quiescent state with a finite number of internal moves. This is the only way the quiescent preorder has to detect a possibility of refusing the performance of output actions due to an internal divergence. In the restricted case above the notion of implementation is represented by the quiescent trace preorder as follows: the condition on the quiescent traces makes sure that, after some stimuli, some output actions will eventually be enabled; the condition on the external traces makes sure that only the desired output actions will be enabled.

The notion above, however, presents some subtle properties. Consider for example

$$A \equiv a \,.\, b \,.\, nil$$

and

$$B \equiv a \,.\, b \,.\, nil + a \,.\, nil$$

where a is an input action and b is an output action. We do not want to consider B as an implementation of A, and the quiescent trace preorder detects the deadlock problem since a is a quiescent trace of B but not a quiescent trace of A. Consider now

$$C \stackrel{\text{def}}{=} c \,.\, C$$

where c is an output action. The result is that

$$C \| B \equiv_Q C \| A.$$

Why does the above result hold? The idea is that, from the point of view of the output actions, the quiescent preorder makes no distinction between the actions of C and those of A. In particular, an output action (c) is always enabled. With the use of the fair preorder the output actions of C are separated from those of A since they constitute two separate classes in the partition of the locally controlled actions of $C \| A$. In the quiescent preorder the partition is constituted by a single class. Notice that the example above is valid also for Receptive Process Theory since C is divergent and the parallel composition of a divergent process with any other process is the divergent process. In other words RPT and the quiescent preorder do not deal with the parallel structure of a system while the fair preorder does.

A new question now arises: Does the quiescent preorder imply the fair preorder in the restricted conditions described above? The answer is "no". Let $X \stackrel{\text{def}}{=} a \,.\, X + b \,.\, X + i \,.\, a \,.\, B$, $B \stackrel{\text{def}}{=} a \,.\, B + b \,.\, B$, $P \stackrel{\text{def}}{=} a \,.\, P' + b \,.\, P'$ and $P' \stackrel{\text{def}}{=} a \,.\, P'$ where a is an input action and b, i are output actions. Then $P \sqsubseteq_Q \tau_{\{i\}}(X)$ but $P \not\sqsubseteq_F \tau_{\{i\}}(X)$ since a^∞ is a fair trace of P but not a fair trace of $\tau_{\{i\}}(X)$. With this example we can also give an example of an intuitive property that is not detected by the quiescent preorder: if the output action b is blocked after n occurrences of action a, then a is not blocked after $n+1$ occurrences of a. The same problem holds also within Receptive Process Theory and within the fair preorder relation. For Receptive Process Theory it is enough to use the same example as above; for the fair preorder it is enough to change the definition of B to $B \stackrel{\text{def}}{=} a \,.\, X + b \,.\, B$ to have the same problem as above with $P \sqsubseteq_F \tau_{\{i\}}(X)$.

The last example presented above is the consequence of a problem that seems general within the field of specification and verification, e.g., the understanding of the actual properties that can be detected by a particular notion of implementation. This topic could be the subject of further research.

6.2 A simple circuit

In this section we use DIOA and the quiescent preorder to specify and verify a simple circuit that is reported in [Jos92]. We start by specifying some simple devices.

A majority element is a device having three input ports and an output one. The voltage level of the output port is that of the majority of the inputs. Every action in the specification represents a change of voltage level in the correspondent port. The process variable M represents the majority element when the voltage levels of its input ports are the same as the voltage level of its output port. The process variables containing subscripts represent the majority element when only the voltage levels of the input ports not appearing as subscripts are the same as the voltage level of the output port. Note that the equation for M_{ab} specifies that no inputs causing a variation in the output voltage level can occur when the output voltage level already has to change. If such inputs occur then the system moves to an unspecified state. Real implementations might actually present glitches on their output ports when such abnormal input sequences occur.

Specification 6.2.1 (majority element) A majority element is specified by the following equations

$$\begin{aligned}
M &\stackrel{\text{def}}{=} a \,.\, M_a + b \,.\, M_b + c \,.\, M_c \\
M_a &\stackrel{\text{def}}{=} a \,.\, M + b \,.\, M_{ab} + c \,.\, M_{ac} \\
M_{ab} &\stackrel{\text{def}}{=} m \,.\, M_c + c \,.\, M_{abc} \\
M_{abc} &\stackrel{\text{def}}{=} m \,.\, M + a \,.\, M_{bc} + b \,.\, M_{ac} + c \,.\, M_{ab}
\end{aligned}$$

where a, b, c are input actions and m is an output action. The equations for M_b, M_c, M_{ac} and M_{bc} are similar to the equations above and can be easily derived.

A wire is simply a device that waits for a change of level in its input port and communicates the change of level through its output port. Input and output actions must be interleaved. If two consecutive inputs are not interleaved with an output then the system moves to the unspecified state.

Specification 6.2.2 (wire) A wire is specified by the following equation:

$$W \stackrel{\text{def}}{=} m \,.\, c \,.\, W$$

where m is an input action and c is an output action.

A Muller element has two inputs and a single output. It waits for a change of level of both its input ports before changing the level of its output port. The subscripts in the process variables represent the input ports that have changed voltage level. When both the inputs have changed (state C_{ab}) the output voltage level is changed.

Specification 6.2.3 (Muller element) A Muller element is specified as follows:

$$\begin{array}{lcl} C & \stackrel{\text{def}}{=} & a \,.\, C_a + b \,.\, C_b \\ C_a & \stackrel{\text{def}}{=} & a \,.\, C + b \,.\, C_{ab} \\ C_b & \stackrel{\text{def}}{=} & a \,.\, C_{ab} + b \,.\, C \\ C_{ab} & \stackrel{\text{def}}{=} & c \,.\, C \end{array}$$

where a, b are input actions and c is an output action.

To give a simple example we formally prove that a Muller element can be implemented using a majority element and a wire.

Proposition 6.2.4 *A Muller element C can be implemented using a majority element and a wire, i.e., $\tau_{\{m\}}(M \| W) \sqsubseteq_Q C$.*

Proof. We show that $\tau_{\{m\}}(M \| W) \sqsubseteq_Q C$. For doing that we consider a family of processes I, I_a, I_b, I_{ab} where $I \stackrel{\text{def}}{=} \tau_{\{m\}}(M \| W)$ and show that they satisfy the equations of C with $\sqsubseteq_Q$. It is then enough to use the recursive substitutivity axiom to conclude.

By applying the expansion axiom and the hiding axioms we obtain

$$
\begin{aligned}
I &\equiv_Q \tau_{\{m\}}(M \| W) && \text{by expanding the process variables} \\
&\equiv_Q \tau_{\{m\}}((a \,.\, M_a + b \,.\, M_b + c \,.\, M_c) \| (m \,.\, c \,.\, W)) && \text{by axiom } \mathbf{E}_2 \\
&\equiv_Q \tau_{\{m\}}(a \,.\, (M_a \| (m \,.\, c \,.\, W)) + b \,.\, (M_b \| (m \,.\, c \,.\, W))) && \text{by substituting } W \text{ for } E(W) \\
&\equiv_Q \tau_{\{m\}}(a \,.\, (M_a \| W) + b \,.\, (M_b \| W)) && \text{by axiom } \mathbf{I}_4 \\
&\equiv_Q \tau_{\{m\}}(a \,.\, (M_a \| W)) + \tau_{\{m\}}(b \,.\, (M_b \| W)) && \text{by axiom } \mathbf{I}_3 \\
&\equiv_Q a \,.\, \tau_{\{m\}}(M_a \| W) + b \,.\, \tau_{\{m\}}(M_b \| W) && \text{by definition of } I_a \text{ and } I_b \\
&\equiv_Q a \,.\, I_a + b \,.\, I_b
\end{aligned}
$$

where we define

$$I_a \stackrel{\text{def}}{=} \tau_{\{m\}}(M_a \| W)$$

$$I_b \stackrel{\text{def}}{=} \tau_{\{m\}}(M_b \| W)$$

With the same method we have

$$I_a \equiv_Q \tau_{\{m\}}(M_a \| W) \equiv_Q a \,.\, \tau_{\{m\}}(M \| W) + b \,.\, \tau_{\{m\}}(M_{ab} \| W) \equiv_Q a \,.\, I + b \,.\, I_{ab}$$

and

$$I_b \equiv_Q \tau_{\{m\}}(M_b \| W) \equiv_Q a \,.\, \tau_{\{m\}}(M_{ab} \| W) + b \,.\, \tau_{\{m\}}(M \| W) \equiv_Q a \,.\, I_{ab} + b \,.\, I$$

where we define

$$I_{ab} \stackrel{\text{def}}{=} \tau_{\{m\}}(M_{ab} \| W)$$

We now proceed with the analysis of I_{ab}. Step by step comments are below.

$$
\begin{aligned}
I_{ab} &\equiv_Q \tau_{\{m\}}(M_{ab} \| W) \\
&\equiv_Q \tau_{\{m\}}(a \,.\, (\Omega \| W) + b \,.\, (\Omega \| W) + m \,.\, (M_c \| c \,.\, W)) \\
&\sqsubseteq_Q \tau_{\{m\}}(m \,.\, (M_c \| c \,.\, W)) \\
&\equiv_Q \tau_{\{m\}}(m \,.\, (a \,.\, (M_{ac} \| c \,.\, W) + b \,.\, (M_{bc} \| c \,.\, W) + c \,.\, (M \| W))) \\
&\sqsubseteq_Q \tau_{\{m\}}(m \,.\, c \,.\, (M \| W)) \\
&\equiv_Q c \,.\, \tau_{\{m\}}(M \| W) \\
&\equiv_Q c \,.\, I
\end{aligned}
$$

The first step follows the lines of the previous derivations by expanding process variables, applying the expansion theorem, and reconverting untouched expanded expressions to their correspondent process variable; the second step is an application of axiom $\mathbf{Ec}_7$ where inputs a and b are eliminated. According to the specification of $C_{a,b}$, in fact, no input should occur before output c occurs. The expression on the second line specifies an implementation choice in the presence of inputs a and b while the expression on the third line does not specify any implementation choice. The third step is similar to the first one while the fourth step consists of successive applications of the hiding axioms. Action m is eliminated through axiom $\mathbf{I}_{11}$ and action c is brought outside the scope of the hiding operator through axiom $\mathbf{I}_3$. The last step is a direct consequence of the definition of I.

We can now apply the recursive substitutivity axiom and conclude. ■

6.3 Handshaking protocol

In this section we use DIOA to specify and verify a circuit realizing the handshaking protocol. The circuit is derived from Kaldewaij [Kal87] and was already specified and verified by means of ACP by Baeten and Vaandrager [BV88]. The main problem encountered in [BV88] is the absence of a distinction between input and output actions in a process. They had to introduce an operator θ to describe the "no output blocking" property of I/O automata and another operator ∇ to limit the traces of a process. In DIOA the "no output blocking" property is granted by the calculus itself, moreover we do not have to restrict the set of traces to consider because the result of giving unespected input actions moves the system to the state Ω from which every trace is admitted. In this way Ω represents the unspecified process, i.e., if the specification of a device moves to Ω for a particular action, then the implementation is correct for whatever behavior it exhibits after performing the same action.

We now give the specifications of some electronic components. A digital component is characterized by a set of input ports and a set of output ports. Each port accepts (or generates) two different signals: HI or LOW. In the rest of this section we will use actions to represent a change of voltage level (from HI to LOW or vice versa) in the signals. In this way, instead of having a pair of actions for each port ($a\uparrow, a\downarrow$) as in [BV88], we have a single action a corresponding to a change of voltage level. We start by specifying an AND port.

Specification 6.3.1 (AND port) The following set of equations specify an AND port.

$$
\begin{aligned}
A^{00}_{xyz} &\stackrel{\text{def}}{=} x \,.\, A^{10}_{xyz} + y \,.\, A^{01}_{xyz}\\
A^{10}_{xyz} &\stackrel{\text{def}}{=} x \,.\, A^{00}_{xyz} + y \,.\, z \,.\, A^{11}_{xyz}\\
A^{01}_{xyz} &\stackrel{\text{def}}{=} x \,.\, z \,.\, A^{11}_{xyz} + y \,.\, A^{00}_{xyz}\\
A^{11}_{xyz} &\stackrel{\text{def}}{=} x \,.\, z \,.\, A^{01}_{xyz} + y \,.\, z \,.\, A^{10}_{xyz}
\end{aligned}
$$

where x, y are input ports and z is an output port. The initial state of the port is $A^{00}_{\bar{x}yz}$ corresponding to both inputs to the low level.

The specification above contains four process variables, each one corresponding to a particular state of the inputs. At each step the port is able to accept an input and consequently change its state. When the output level has to change it is not permitted sending other input until the output level is changed. An input action sent while the system is changing its output state will move the system to an unspecified state. The next specification introduces an AND port with a negated input. The line under x specifies that port x is negated.

Specification 6.3.2 (AND port with a negated input) The following equations specify an AND port with a negated input.

$$
\begin{aligned}
A^{00}_{\underline{x}yz} &\stackrel{\text{def}}{=} x \,.\, A^{10}_{\underline{x}yz} + y \,.\, A^{01}_{\underline{x}yz}\\
A^{10}_{\underline{x}yz} &\stackrel{\text{def}}{=} x \,.\, A^{00}_{\underline{x}yz} + y \,.\, z \,.\, A^{11}_{\underline{x}yz}\\
A^{01}_{\underline{x}yz} &\stackrel{\text{def}}{=} x \,.\, z \,.\, A^{11}_{\underline{x}yz} + y \,.\, A^{00}_{\underline{x}yz}\\
A^{11}_{\underline{x}yz} &\stackrel{\text{def}}{=} x \,.\, z \,.\, A^{01}_{\underline{x}yz} + y \,.\, z \,.\, A^{10}_{\underline{x}yz}
\end{aligned}
$$

where x, y are input actions and z is an output action. The initial state of the port is $A^{10}_{\bar{x}yz}$ corresponding to both inputs to the low level.

The AND port with a negated input is identical to the AND port with the difference that the output signal changes in different points (in the above specification the initial state is different from the initial state of specification 6.3.1). Note that, by opportunely renaming the process variables, we can obtain the specification of the AND port. Another interesting observation is that, after giving the specification of an inverter (a component giving as output the opposite of its input), the AND port with a negated input can not be implemented using a normal AND

Figure 6-1: Symbolic representation of AND ports

port with an inverter. In fact its correctness strictly depends on time assumptions about the occurrences of new inputs and the speed of the components. The two kinds of AND ports we have just introduced are represented in figure 6-1. We proceed by specifying a Muller C element. The specification below is similar to the one given in the previous section: it gives more restrictions to the occurrences of input actions. A Muller C element is essentially a component that waits for the change of both its input levels and then changes its output. Every input port can not be changed more then once between one change in the output and the successive one.

Specification 6.3.3 (Muller C element) The following equations specify a Muller element.

$$C^0_{xyz} \stackrel{\text{def}}{=} x \,.\, y \,.\, z \,.\, C^1_{xyz} + y \,.\, x \,.\, z \,.\, C^1_{xyz}$$
$$C^1_{xyz} \stackrel{\text{def}}{=} x \,.\, y \,.\, z \,.\, C^0_{xyz} + y \,.\, x \,.\, z \,.\, C^0_{xyz}$$

where x, y are input actions and z is an output action. The initial state of the process is $C^0_{\underline{x}yz}$ corresponding to all the interfaces to the low level.

The following specification introduces a Muller C element with a negated input. It is immediate to observe that the only difference from the normal Muller element is in the initial state. This is because we use actions to represent only changes of level and not the kind of variation itself.

Specification 6.3.4 (Muller element with a negated input) The following equations specify a Muller element with a negated input.

$$C^0_{\underline{x}yz} \stackrel{\text{def}}{=} x \,.\, y \,.\, z \,.\, C^1_{\underline{x}yz} + y \,.\, x \,.\, z \,.\, C^1_{\underline{x}yz}$$
$$C^1_{\underline{x}yz} \stackrel{\text{def}}{=} x \,.\, y \,.\, z \,.\, C^0_{\underline{x}yz} + y \,.\, x \,.\, z \,.\, C^0_{\underline{x}yz}$$

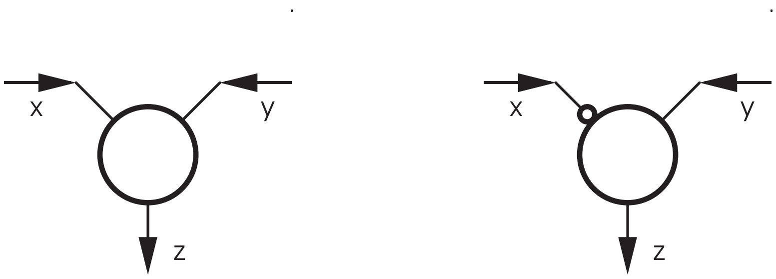

Figure 6-2: Symbolic representation of Muller elements

where x, y are input actions and z is an output action. The initial state of the process is $y \,.\, z \,.\, C^1_{\underline{x}yz}$ corresponding to all the interfaces to the low level.

Figure 6-2 represents the two kinds of Muller elements introduced above.

We are now ready to specify the handshaking bit protocol. This protocol is often used to avoid interference between circuits. The circuit has two input wires a, b and two output wires $\bar{a}, \bar{b}$. It has to follow the four-phase handshaking protocol for the pairs $a, \bar{a}$ and $b, \bar{b}$ where $a, \bar{a}$ is the input side. This means that on the input side an external process will change the level of a and wait for a change of $\bar{a}$ and then repeat the same process; on the other side the output process waits for a change in action $\bar{b}$ and changes the level of b. It then repeats this pair of actions. No other kinds of interactions are admitted for the protocol. For example changing the level of a twice without waiting for the change of $\bar{a}$ will move the system to an unspecified state.

Specification 6.3.5 (handshaking protocol) The following equations specify the handshaking protocol.

$$S \stackrel{\text{def}}{=} a \,.\, S^*$$
$$S^* \stackrel{\text{def}}{=} \bar{a} \,.\, a \,.\, \bar{a} \,.\, S^*_1$$
$$S^*_1 \stackrel{\text{def}}{=} \bar{b} \,.\, S^*_2 + a \,.\, \bar{b} \,.\, b \,.\, \bar{b} \,.\, b \,.\, S^*$$
$$S^*_2 \stackrel{\text{def}}{=} b \,.\, S^*_3 + a \,.\, b \,.\, \bar{b} \,.\, b \,.\, S^*$$
$$S^*_3 \stackrel{\text{def}}{=} \bar{b} \,.\, S^*_4 + a \,.\, \bar{b} \,.\, b \,.\, S^*$$
$$S^*_4 \stackrel{\text{def}}{=} b \,.\, S + a \,.\, b \,.\, S^*$$

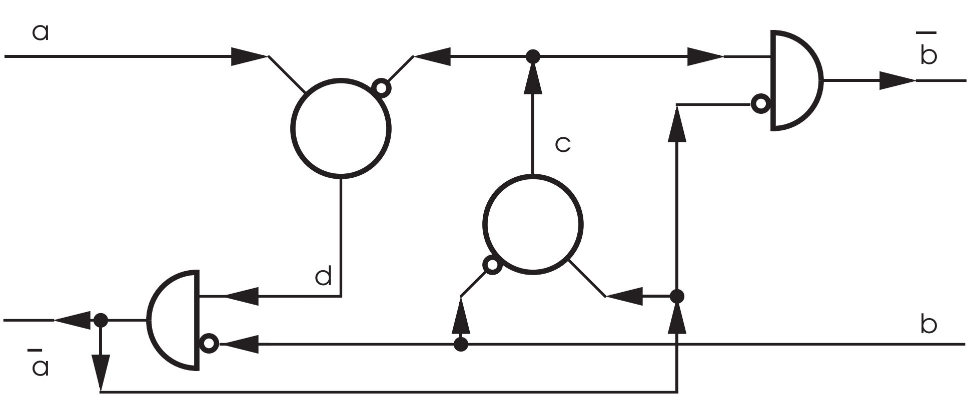

Figure 6-3: Implementation of the handshaking protocol

where a, b are input actions and $\bar{a}, \bar{b}$ are output actions. The initial state is S.

We now propose an implementation in which we assume instantaneous communication between the components. This is a simplification of the implementation given in [BV88]. The implementation is the following process M.

$$M \stackrel{\text{def}}{=} \tau_{\{c,d\}}(a \,.\, d \,.\, C^{1}_{\underline{c}ad} \| A^{10}_{\underline{b}d\bar{a}} \| \bar{a} \,.\, c \,.\, C^{1}_{\underline{b}\bar{a}c} \| A^{10}_{\underline{\bar{a}}c\bar{b}})$$

In the following we will let $H = \{c, d\}$. It is immediate to verify that M can not diverge since every component having the control of an internal action must perform an external action before completing a cycle. Figure 6-3 represents process M. We proceed by giving the proof of correctness.

Proposition 6.3.6 (correctness of M**)** *The implementation of the buffer is correct. In other words* $M \sqsubseteq_Q S$.

Proof. To prove the correctness of the implementation we find a set of expressions

$$\tilde{M} = \{M, M^*, M^*_1, M^*_2, M^*_3, M^*_4\}$$

that satisfies $\tilde{M} \sqsubseteq_Q \tilde{E}(\tilde{S})[\tilde{M}/\tilde{S}]$. In this way we can apply the recursive substitutivity axiom to conclude. To prove the equations we continuously perform steps by means of the expansion axiom and then eliminate (if possible) undesired actions. We start by considering process M.

$$
\begin{aligned}
M &\equiv_Q \tau_H(a\,.\,(d\,.\,C^1_{\underline{c}ad} \| A^{10}_{\underline{b}d\bar{a}} \| \bar{a}\,.\,c\,.\,C^1_{\underline{b}\bar{a}c} \| A^{10}_{\underline{\bar{a}}c\bar{b}}) + b\,.\,(a\,.\,d\,.\,C^1_{\underline{c}ad} \| A^{00}_{\underline{b}d\bar{a}} \| \Omega \| A^{10}_{\underline{\bar{a}}c\bar{b}})) \\
&\sqsubseteq_Q \tau_H(a\,.\,(d\,.\,C^1_{\underline{c}ad} \| A^{10}_{\underline{b}d\bar{a}} \| \bar{a}\,.\,c\,.\,C^1_{\underline{b}\bar{a}c} \| A^{10}_{\underline{\bar{a}}c\bar{b}})) \\
&\equiv_Q a\,.\,\tau_H(d\,.\,C^1_{\underline{c}ad} \| A^{10}_{\underline{b}d\bar{a}} \| \bar{a}\,.\,c\,.\,C^1_{\underline{b}\bar{a}c} \| A^{10}_{\underline{\bar{a}}c\bar{b}}) \\
&\equiv_Q a\,.\,\tau_H(d\,.\,(C^1_{\underline{c}ad} \| \bar{a}\,.\,A^{11}_{\underline{b}d\bar{a}} \| \bar{a}\,.\,c\,.\,C^1_{\underline{b}\bar{a}c} \| A^{10}_{\underline{\bar{a}}c\bar{b}}) + a\,.\,(\Omega \| A^{10}_{\underline{b}d\bar{a}} \| \bar{a}\,.\,c\,.\,C^1_{\underline{b}\bar{a}c} \| A^{10}_{\underline{\bar{a}}c\bar{b}}) + \\
&\qquad b\,.\,(d\,.\,C^1_{\underline{c}ad} \| A^{00}_{\underline{b}d\bar{a}} \| \Omega \| A^{10}_{\underline{\bar{a}}c\bar{b}})) \\
&\sqsubseteq_Q a\,.\,\tau_H(d\,.\,(C^1_{\underline{c}ad} \| \bar{a}\,.\,A^{11}_{\underline{b}d\bar{a}} \| \bar{a}\,.\,c\,.\,C^1_{\underline{b}\bar{a}c} \| A^{10}_{\underline{\bar{a}}c\bar{b}}))
\end{aligned}
$$

In the first step we have applied the expansion axiom together with the substitutivity axiom for the hiding operator. To obtain the expression above we implicitly assume that the application of the expansion axiom proceeds as follows: unfold process variables that are not prefixed, apply the expansion axiom, fold unchanged unfolded expressions. Since we are not interested in the effects of action b (the equation for S does not consider action b) we use axiom $\mathbf{Ec}_7$ in the second step to eliminate the summand prefixed by b. In the third step we use axiom $\mathbf{I}_3$ to move the prefix a outside the hiding operator. We then apply the expansion axiom again and eliminate the undesired input actions with axiom $\mathbf{Ec}_7$ in the following two steps. Note that we choose the input actions to eliminate by looking at the specification 6.3.5. It is clear, in fact, that at this stage we do not have to wait for any input action until action $\bar{a}$ is performed. If an input action occurs before action $\bar{a}$ is performed then any behavior is admissible.

In the last step we have an internal action d. In order to eliminate this action we have to substitute its prefixed expression with an expression for which axiom $\mathbf{I}_{11}$ is appliable. For this reason let

$$M' \stackrel{\text{def}}{=} C^1_{\underline{c}ad} \| \bar{a}\,.\,A^{11}_{\underline{b}d\bar{a}} \| \bar{a}\,.\,c\,.\,C^1_{\underline{b}\bar{a}c} \| A^{10}_{\underline{\bar{a}}c\bar{b}}$$

By using the expansion axiom for the first step and axiom $\mathbf{Ec}_7$ to eliminate undesired inputs we have

$$
\begin{aligned}
\text{M'} &\equiv_Q \bar{a}\,.\,(C^1_{\underline{c}ad} \| A^{11}_{\underline{b}d\bar{a}} \| c\,.\,C^1_{\underline{b}\bar{a}c} \| A^{00}_{\underline{\bar{a}}c\bar{b}}) + b\,.\,(C^1_{\underline{c}ad} \| \Omega \| \Omega \| A^{10}_{\underline{\bar{a}}c\bar{b}}) + \\
&\qquad a\,.\,(c\,.\,d\,.\,C^1_{\underline{c}ad} \| \bar{a}\,.\,A^{11}_{\underline{b}d\bar{a}} \| \bar{a}\,.\,c\,.\,C^1_{\underline{b}\bar{a}c} \| A^{10}_{\underline{\bar{a}}c\bar{b}}) \\
&\sqsubseteq_Q \bar{a}\,.\,(C^1_{\underline{c}ad} \| A^{11}_{\underline{b}d\bar{a}} \| c\,.\,C^1_{\underline{b}\bar{a}c} \| A^{00}_{\underline{\bar{a}}c\bar{b}})
\end{aligned}
$$

By substituting this last expression in the last expression obtained from M we have

$$M \sqsubseteq_Q a\,.\,\tau_H(d\,.\,\bar{a}\,.\,(C^1_{\underline{c}ad} \| A^{11}_{\underline{b}d\bar{a}} \| c\,.\,C^1_{\underline{b}\bar{a}c} \| A^{00}_{\underline{\bar{a}}c\bar{b}}))$$

$$\equiv_Q a\,.\,\tau_H(\bar{a}\,.\,(C^1_{\underline{c}ad}\|A^{11}_{\underline{b}d\bar{a}}\|c\,.\,C^1_{\underline{b}\bar{a}c}\|A^{00}_{\underline{\bar{a}}c\bar{b}}))$$

where the last expression is obtained by means of axiom $\mathbf{I}_{11}$. Note in fact that $wsi(\bar{a}\,.\,(C^1_{\underline{c}ad}\|A^{11}_{\underline{b}d\bar{a}}\|c\,.\,C^1_{\underline{b}\bar{a}c}\|A^{00}_{\underline{\bar{a}}c\bar{b}})) = \emptyset$. Let

$$M^* \stackrel{\text{def}}{=} \tau_H(\bar{a}\,.\,(C^1_{\underline{c}ad}\|A^{11}_{\underline{b}d\bar{a}}\|c\,.\,C^1_{\underline{b}\bar{a}c}\|A^{00}_{\underline{\bar{a}}c\bar{b}}))$$

We have just shown that

$$M \sqsubseteq_Q a\,.\,M^*$$

We now proceed on the analysis of M^*. Since we often have to eliminate undesired input actions we use the convention of not writing expressions that have to be eliminated in subsequent steps. This convention is immediately clear from the following steps.

$$\begin{aligned}
M^* &\equiv_Q \bar{a}\,.\,\tau_H(C^1_{\underline{c}ad}\|A^{11}_{\underline{b}d\bar{a}}\|c\,.\,C^1_{\underline{b}\bar{a}c}\|A^{00}_{\underline{\bar{a}}c\bar{b}})\\
&\equiv_Q \bar{a}\,.\,\tau_H(a\,.\,(c\,.\,d\,.\,C^0_{\underline{c}ad}\|A^{11}_{\underline{b}d\bar{a}}\|c\,.\,C^1_{\underline{b}\bar{a}c}\|A^{00}_{\underline{\bar{a}}c\bar{b}}) + b\,.\,(\ldots) +\\
&\qquad c\,.\,(a\,.\,d\,.\,C^0_{\underline{c}ad}\|A^{11}_{\underline{b}d\bar{a}}\|C^1_{\underline{b}\bar{a}c}\|A^{01}_{\underline{\bar{a}}c\bar{b}}))\\
&\sqsubseteq_Q \bar{a}\,.\,\tau_H(a\,.\,(c\,.\,d\,.\,C^0_{\underline{c}ad}\|A^{11}_{\underline{b}d\bar{a}}\|c\,.\,C^1_{\underline{b}\bar{a}c}\|A^{00}_{\underline{\bar{a}}c\bar{b}}) + c\,.\,(a\,.\,d\,.\,C^0_{\underline{c}ad}\|A^{11}_{\underline{b}d\bar{a}}\|C^1_{\underline{b}\bar{a}c}\|A^{01}_{\underline{\bar{a}}c\bar{b}}))
\end{aligned}$$

In the previous steps we again used the expansion axiom together with axiom $\mathbf{Ec}_7$. At this point we can not proceed without solving the most internal expressions because there is an internal action as prefix. We then simplify the internal expressions as follows:

$$\begin{aligned}
&c\,.\,d\,.\,C^0_{\underline{c}ad}\|A^{11}_{\underline{b}d\bar{a}}\|c\,.\,C^1_{\underline{b}\bar{a}c}\|A^{00}_{\underline{\bar{a}}c\bar{b}}\\
\equiv_Q\ &c\,.\,(d\,.\,C^0_{\underline{c}ad}\|A^{11}_{\underline{b}d\bar{a}}\|C^1_{\underline{b}\bar{a}c}\|A^{01}_{\underline{\bar{a}}c\bar{b}}) + a\,.\,(\ldots) + b\,.\,(\ldots)\\
\sqsubseteq_Q\ &c\,.\,(d\,.\,C^0_{\underline{c}ad}\|A^{11}_{\underline{b}d\bar{a}}\|C^1_{\underline{b}\bar{a}c}\|A^{01}_{\underline{\bar{a}}c\bar{b}})
\end{aligned}$$

$$\begin{aligned}
&a\,.\,d\,.\,C^0_{\underline{c}ad}\|A^{11}_{\underline{b}d\bar{a}}\|C^1_{\underline{b}\bar{a}c}\|A^{01}_{\underline{\bar{a}}c\bar{b}}\\
\equiv_Q\ &a\,.\,(d\,.\,C^0_{\underline{c}ad}\|A^{11}_{\underline{b}d\bar{a}}\|C^1_{\underline{b}\bar{a}c}\|A^{01}_{\underline{\bar{a}}c\bar{b}}) + b\,.\,(\ldots)\\
\sqsubseteq_Q\ &a\,.\,(d\,.\,C^0_{\underline{c}ad}\|A^{11}_{\underline{b}d\bar{a}}\|C^1_{\underline{b}\bar{a}c}\|A^{01}_{\underline{\bar{a}}c\bar{b}})
\end{aligned}$$

where we have again used the expansion axiom and axiom $\mathbf{Ec}_7$. By combining the last two inequalities with the last expression obtained for M^* we have

$$
\begin{aligned}
M^* &\sqsubseteq_Q \bar{a} \,.\, \tau_H(a \,.\, (c \,.\, d \,.\, C^0_{\underline{c}ad} \| A^{11}_{\underline{b}d\bar{a}} \| c \,.\, C^1_{\underline{b}\bar{a}c} \| A^{00}_{\underline{\bar{a}}c\bar{b}}) + c \,.\, (a \,.\, d \,.\, C^0_{\underline{c}ad} \| A^{11}_{\underline{b}d\bar{a}} \| C^1_{\underline{b}\bar{a}c} \| A^{01}_{\underline{\bar{a}}c\bar{b}})) \\
&\sqsubseteq_Q \bar{a} \,.\, \tau_H(a \,.\, c \,.\, (d \,.\, C^0_{\underline{c}ad} \| A^{11}_{\underline{b}d\bar{a}} \| C^1_{\underline{b}\bar{a}c} \| A^{01}_{\underline{\bar{a}}c\bar{b}}) + c \,.\, a \,.\, (d \,.\, C^0_{\underline{c}ad} \| A^{11}_{\underline{b}d\bar{a}} \| C^1_{\underline{b}\bar{a}c} \| A^{01}_{\underline{\bar{a}}c\bar{b}})
\end{aligned}
$$

We now have to apply axiom $\mathbf{I}_{16}$, but we first have to simplify the internal expression in order to satisfy the condition for axiom $\mathbf{I}_{16}$.

$$
\begin{aligned}
& d \,.\, C^0_{\underline{c}ad} \| A^{11}_{\underline{b}d\bar{a}} \| C^1_{\underline{b}\bar{a}c} \| A^{01}_{\underline{\bar{a}}c\bar{b}} \\
\equiv_Q \; & d \,.\, (C^0_{\underline{c}ad} \| \bar{a} \,.\, A^{10}_{\underline{b}d\bar{a}} \| C^1_{\underline{b}\bar{a}c} \| A^{01}_{\underline{\bar{a}}c\bar{b}}) + a \,.\, (\ldots) + b \,.\, (\ldots) \\
\sqsubseteq_Q \; & d \,.\, (C^0_{\underline{c}ad} \| \bar{a} \,.\, A^{10}_{\underline{b}d\bar{a}} \| C^1_{\underline{b}\bar{a}c} \| A^{01}_{\underline{\bar{a}}c\bar{b}})
\end{aligned}
$$

By substituting in the last expression for M^*

$$
\begin{aligned}
M^* &\sqsubseteq_Q \bar{a} \,.\, \tau_H(a \,.\, c \,.\, d \,.\, (C^0_{\underline{c}ad} \| \bar{a} \,.\, A^{10}_{\underline{b}d\bar{a}} \| C^1_{\underline{b}\bar{a}c} \| A^{01}_{\underline{\bar{a}}c\bar{b}}) + c \,.\, a \,.\, d \,.\, (C^0_{\underline{c}ad} \| \bar{a} \,.\, A^{10}_{\underline{b}d\bar{a}} \| C^1_{\underline{b}\bar{a}c} \| A^{01}_{\underline{\bar{a}}c\bar{b}})) \\
&\equiv_Q \bar{a} \,.\, \tau_H(a \,.\, d \,.\, (C^0_{\underline{c}ad} \| \bar{a} \,.\, A^{10}_{\underline{b}d\bar{a}} \| C^1_{\underline{b}\bar{a}c} \| A^{01}_{\underline{\bar{a}}c\bar{b}})) \\
&\equiv_Q \bar{a} \,.\, a \,.\, \tau_H(d \,.\, (C^0_{\underline{c}ad} \| \bar{a} \,.\, A^{10}_{\underline{b}d\bar{a}} \| C^1_{\underline{b}\bar{a}c} \| A^{01}_{\underline{\bar{a}}c\bar{b}})) \\
&\equiv_Q \bar{a} \,.\, a \,.\, \tau_H(d \,.\, (\bar{a} \,.\, (C^0_{\underline{c}ad} \| A^{10}_{\underline{b}d\bar{a}} \| b \,.\, c \,.\, C^0_{\underline{b}\bar{a}c} \| \bar{b} \,.\, A^{11}_{\underline{\bar{a}}c\bar{b}}) + a \,.\, (\ldots) + b \,.\, (\ldots))) \\
&\sqsubseteq_Q \bar{a} \,.\, a \,.\, \tau_H(d \,.\, \bar{a} \,.\, (C^0_{\underline{c}ad} \| A^{10}_{\underline{b}d\bar{a}} \| b \,.\, c \,.\, C^0_{\underline{b}\bar{a}c} \| \bar{b} \,.\, A^{11}_{\underline{\bar{a}}c\bar{b}})) \\
&\equiv_Q \bar{a} \,.\, a \,.\, \tau_H(\bar{a} \,.\, (C^0_{\underline{c}ad} \| A^{10}_{\underline{b}d\bar{a}} \| b \,.\, c \,.\, C^0_{\underline{b}\bar{a}c} \| \bar{b} \,.\, A^{11}_{\underline{\bar{a}}c\bar{b}})) \\
&\equiv_Q \bar{a} \,.\, a \,.\, \bar{a} \,.\, \tau_H(C^0_{\underline{c}ad} \| A^{10}_{\underline{b}d\bar{a}} \| b \,.\, c \,.\, C^0_{\underline{b}\bar{a}c} \| \bar{b} \,.\, A^{11}_{\underline{\bar{a}}c\bar{b}})
\end{aligned}
$$

In the first step we used axiom $\mathbf{I}_{16}$. The rest of the steps are obtained by using the expansion axiom together with axiom $\mathbf{Ec}_7$ (an the substitutivity rules of course). The last but one step is obtained using axiom $\mathbf{I}_{11}$. We can now define the new process

$$
M_1^* \stackrel{\text{def}}{=} \tau_H(C^0_{\underline{c}ad} \| A^{10}_{\underline{b}d\bar{a}} \| b \,.\, c \,.\, C^0_{\underline{b}\bar{a}c} \| \bar{b} \,.\, A^{11}_{\underline{\bar{a}}c\bar{b}})
$$

What we have just shown is

$$
M^* \sqsubseteq_Q \bar{a} \,.\, a \,.\, \bar{a} \,.\, M_1^*
$$

The following simplifications are new only for the third step. In this case we use axiom $\mathbf{I}_4$ followed by axiom $\mathbf{I}_3$.

$$
\begin{aligned}
M_1^* \equiv_Q \; & \tau_H(\bar{b} \,.\, (C^0_{\underline{c}ad} \| A^{10}_{\underline{b}d\bar{a}} \| b \,.\, c \,.\, C^0_{\underline{b}\bar{a}c} \| A^{11}_{\underline{\bar{a}}c\bar{b}}) + b \,.\, (\ldots) + \\
& a \,.\, (c \,.\, C^1_{\underline{c}ad} \| A^{10}_{\underline{b}d\bar{a}} \| b \,.\, c \,.\, C^0_{\underline{b}\bar{a}c} \| \bar{b} \,.\, A^{11}_{\underline{\bar{a}}c\bar{b}}))
\end{aligned}
$$

$$
\begin{aligned}
&\sqsubseteq_Q \tau_H(\bar{b} \,.\, (C^0_{\underline{c}ad} \| A^{10}_{\underline{b}d\bar{a}} \| b \,.\, c \,.\, C^0_{\underline{b}\bar{a}c} \| A^{11}_{\underline{\bar{a}}c\bar{b}}) + a \,.\, (c \,.\, C^1_{\underline{c}ad} \| A^{10}_{\underline{b}d\bar{a}} \| b \,.\, c \,.\, C^0_{\underline{b}\bar{a}c} \| \bar{b} \,.\, A^{11}_{\underline{\bar{a}}c\bar{b}}))\\
&\equiv_Q \bar{b} \,.\, \tau_H(C^0_{\underline{c}ad} \| A^{10}_{\underline{b}d\bar{a}} \| b \,.\, c \,.\, C^0_{\underline{b}\bar{a}c} \| A^{11}_{\underline{\bar{a}}c\bar{b}}) + a \,.\, \tau_H(c \,.\, d \,.\, C^1_{\underline{c}ad} \| A^{10}_{\underline{b}d\bar{a}} \| b \,.\, c \,.\, C^0_{\underline{b}\bar{a}c} \| \bar{b} \,.\, A^{11}_{\underline{\bar{a}}c\bar{b}})\\
&\equiv_Q \bar{b} \,.\, \tau_H(C^0_{\underline{c}ad} \| A^{10}_{\underline{b}d\bar{a}} \| b \,.\, c \,.\, C^0_{\underline{b}\bar{a}c} \| A^{11}_{\underline{\bar{a}}c\bar{b}}) +\\
&\qquad a \,.\, \tau_H(\bar{b} \,.\, (c \,.\, d \,.\, C^1_{\underline{c}ad} \| A^{10}_{\underline{b}d\bar{a}} \| b \,.\, c \,.\, C^0_{\underline{b}\bar{a}c} \| A^{11}_{\underline{\bar{a}}c\bar{b}}) + a \,.\, (\ldots) + b \,.\, (\ldots))\\
&\sqsubseteq_Q \bar{b} \,.\, \tau_H(C^0_{\underline{c}ad} \| A^{10}_{\underline{b}d\bar{a}} \| b \,.\, c \,.\, C^0_{\underline{b}\bar{a}c} \| A^{11}_{\underline{\bar{a}}c\bar{b}}) + a \,.\, \tau_H(\bar{b} \,.\, (c \,.\, d \,.\, C^1_{\underline{c}ad} \| A^{10}_{\underline{b}d\bar{a}} \| b \,.\, c \,.\, C^0_{\underline{b}\bar{a}c} \| A^{11}_{\underline{\bar{a}}c\bar{b}}))\\
&\equiv_Q \bar{b} \,.\, \tau_H(C^0_{\underline{c}ad} \| A^{10}_{\underline{b}d\bar{a}} \| b \,.\, c \,.\, C^0_{\underline{b}\bar{a}c} \| A^{11}_{\underline{\bar{a}}c\bar{b}}) + a \,.\, \bar{b} \,.\, \tau_H(c \,.\, d \,.\, C^1_{\underline{c}ad} \| A^{10}_{\underline{b}d\bar{a}} \| b \,.\, c \,.\, C^0_{\underline{b}\bar{a}c} \| A^{11}_{\underline{\bar{a}}c\bar{b}})
\end{aligned}
$$

We now define two new processes:

$$M_2^* \stackrel{\text{def}}{=} \tau_H(C^0_{\underline{c}ad} \| A^{10}_{\underline{b}d\bar{a}} \| b \,.\, c \,.\, C^0_{\underline{b}\bar{a}c} \| A^{11}_{\underline{\bar{a}}c\bar{b}})$$

$$M_1 \stackrel{\text{def}}{=} \tau_H(c \,.\, d \,.\, C^1_{\underline{c}ad} \| A^{10}_{\underline{b}d\bar{a}} \| b \,.\, c \,.\, C^0_{\underline{b}\bar{a}c} \| A^{11}_{\underline{\bar{a}}c\bar{b}})$$

What we have just shown is

$$M_1^* \sqsubseteq_Q \bar{b} \,.\, M_2^* + a \,.\, \bar{b} \,.\, M_1$$

We start by analyzing M_1.

$$
\begin{aligned}
M_1 &\equiv_Q \tau_H(a \,.\, (\ldots) + b \,.\, (c \,.\, d \,.\, C^1_{\underline{c}ad} \| A^{00}_{\underline{b}d\bar{a}} \| c \,.\, C^0_{\underline{b}\bar{a}c} \| A^{11}_{\underline{\bar{a}}c\bar{b}}))\\
&\sqsubseteq_Q \tau_H(b \,.\, (c \,.\, d \,.\, C^1_{\underline{c}ad} \| A^{00}_{\underline{b}d\bar{a}} \| c \,.\, C^0_{\underline{b}\bar{a}c} \| A^{11}_{\underline{\bar{a}}c\bar{b}}))\\
&\equiv_Q b \,.\, \tau_H(c \,.\, d \,.\, C^1_{\underline{c}ad} \| A^{00}_{\underline{b}d\bar{a}} \| c \,.\, C^0_{\underline{b}\bar{a}c} \| A^{11}_{\underline{\bar{a}}c\bar{b}})\\
&\equiv_Q b \,.\, \tau_H(c \,.\, (d \,.\, C^1_{\underline{c}ad} \| A^{00}_{\underline{b}d\bar{a}} \| C^0_{\underline{b}\bar{a}c} \| \bar{b} \,.\, A^{10}_{\underline{\bar{a}}c\bar{b}}) +\\
&\qquad a \,.\, (\ldots) + b \,.\, (\ldots))\\
&\sqsubseteq_Q b \,.\, \tau_H(c \,.\, (d \,.\, C^1_{\underline{c}ad} \| A^{00}_{\underline{b}d\bar{a}} \| C^0_{\underline{b}\bar{a}c} \| \bar{b} \,.\, A^{10}_{\underline{\bar{a}}c\bar{b}}))
\end{aligned}
$$

The steps above are again the application of the expansion axiom and axioms $\mathbf{Ec}_7$ and $\mathbf{I}_3$. However it is not possible for the moment to eliminate the internal prefix c because we first have to simplify its prefixed expression. We then define

$$M_2 \stackrel{\text{def}}{=} \tau_H(d \,.\, C^1_{\underline{c}ad} \| A^{00}_{\underline{b}d\bar{a}} \| C^0_{\underline{b}\bar{a}c} \| \bar{b} \,.\, A^{10}_{\underline{\bar{a}}c\bar{b}})$$

simplify M_2, and then substitute the result in the last expression for M_1 by means of axiom $\mathbf{I}_{11}$.

$$
\begin{aligned}
M_2 \equiv_Q\ & \tau_H(d\,.\,(C^1_{\underline{c}ad}\|A^{01}_{\underline{b}d\bar{a}}\|C^0_{\underline{b}\bar{a}c}\|\bar{b}\,.\,A^{10}_{\underline{\bar{a}}c\bar{b}} + a\,.\,(\ldots)+\\
& \bar{b}\,.\,(d\,.\,C^1_{\underline{c}ad}\|A^{00}_{\underline{b}d\bar{a}}\|C^0_{\underline{b}\bar{a}c}\|A^{10}_{\underline{\bar{a}}c\bar{b}}) + b\,.\,(\ldots))\\
\sqsubseteq_Q\ & \tau_H(d\,.\,(C^1_{\underline{c}ad}\|A^{01}_{\underline{b}d\bar{a}}\|C^0_{\underline{b}\bar{a}c}\|\bar{b}\,.\,A^{10}_{\underline{\bar{a}}c\bar{b}} + \bar{b}\,.\,(d\,.\,C^1_{\underline{c}ad}\|A^{00}_{\underline{b}d\bar{a}}\|C^0_{\underline{b}\bar{a}c}\|A^{10}_{\underline{\bar{a}}c\bar{b}}))\\
\equiv_Q\ & \tau_H(d\,.\,(\bar{b}\,.\,(C^1_{\underline{c}ad}\|A^{01}_{\underline{b}d\bar{a}}\|C^0_{\underline{b}\bar{a}c}\|A^{10}_{\underline{\bar{a}}c\bar{b}}) + a\,.\,(\ldots) + b\,.\,(\ldots))+\\
& \bar{b}\,.\,(d\,.\,(C^1_{\underline{c}ad}\|A^{01}_{\underline{b}d\bar{a}}\|C^0_{\underline{b}\bar{a}c}\|A^{10}_{\underline{\bar{a}}c\bar{b}}) + a\,.\,(\ldots) + b\,.\,(d\,.\,C^1_{\underline{c}ad}\|A^{10}_{\underline{b}d\bar{a}}\|\bar{a}\,.\,c\,.\,C^1_{\underline{b}\bar{a}c}\|A^{10}_{\underline{\bar{a}}c\bar{b}})))\\
\sqsubseteq_Q\ & \tau_H(d\,.\,\bar{b}\,.\,(C^1_{\underline{c}ad}\|A^{01}_{\underline{b}d\bar{a}}\|C^0_{\underline{b}\bar{a}c}\|A^{10}_{\underline{\bar{a}}c\bar{b}})+\\
& \bar{b}\,.\,(d\,.\,(C^1_{\underline{c}ad}\|A^{01}_{\underline{b}d\bar{a}}\|C^0_{\underline{b}\bar{a}c}\|A^{10}_{\underline{\bar{a}}c\bar{b}}) + b\,.\,(d\,.\,C^1_{\underline{c}ad}\|A^{10}_{\underline{b}d\bar{a}}\|\bar{a}\,.\,c\,.\,C^1_{\underline{b}\bar{a}c}\|A^{10}_{\underline{\bar{a}}c\bar{b}})))
\end{aligned}
$$

The steps above are again standard. Note however that in the third step we have to accept the input action b in the expression prefixed by b. This follows from the specification of S_1^*. We will show later that failing to accept action b will generate an error. To proceed we first have to simplify the internal expressions.

$$
\begin{aligned}
& C^1_{\underline{c}ad}\|A^{01}_{\underline{b}d\bar{a}}\|C^0_{\underline{b}\bar{a}c}\|A^{10}_{\underline{\bar{a}}c\bar{b}}\\
\equiv_Q\ & a\,.\,(\ldots) + b\,.\,(C^1_{\underline{c}ad}\|\bar{a}\,.\,A^{11}_{\underline{b}d\bar{a}}\|\bar{a}\,.\,c\,.\,C^1_{\underline{b}\bar{a}c}\|A^{10}_{\underline{\bar{a}}c\bar{b}})\\
\sqsubseteq_Q\ & b\,.\,(C^1_{\underline{c}ad}\|\bar{a}\,.\,A^{11}_{\underline{b}d\bar{a}}\|\bar{a}\,.\,c\,.\,C^1_{\underline{b}\bar{a}c}\|A^{10}_{\underline{\bar{a}}c\bar{b}})
\end{aligned}
$$

$$
\begin{aligned}
& d\,.\,C^1_{\underline{c}ad}\|A^{10}_{\underline{b}d\bar{a}}\|\bar{a}\,.\,c\,.\,C^1_{\underline{b}\bar{a}c}\|A^{10}_{\underline{\bar{a}}c\bar{b}}\\
\equiv_Q\ & d\,.\,(C^1_{\underline{c}ad}\|\bar{a}\,.\,A^{11}_{\underline{b}d\bar{a}}\|\bar{a}\,.\,c\,.\,C^1_{\underline{b}\bar{a}c}\|A^{00}_{\underline{\bar{a}}c\bar{b}}) + a\,.\,(\ldots) + b\,.\,(\ldots)\\
\sqsubseteq_Q\ & d\,.\,(C^1_{\underline{c}ad}\|\bar{a}\,.\,A^{11}_{\underline{b}d\bar{a}}\|\bar{a}\,.\,c\,.\,C^1_{\underline{b}\bar{a}c}\|A^{00}_{\underline{\bar{a}}c\bar{b}})
\end{aligned}
$$

$$
\begin{aligned}
& C^1_{\underline{c}ad}\|\bar{a}\,.\,A^{11}_{\underline{b}d\bar{a}}\|\bar{a}\,.\,c\,.\,C^1_{\underline{b}\bar{a}c}\|A^{00}_{\underline{\bar{a}}c\bar{b}}\\
\equiv_Q\ & \bar{a}\,.\,(C^1_{\underline{c}ad}\|A^{11}_{\underline{b}d\bar{a}}\|c\,.\,C^1_{\underline{b}\bar{a}c}\|A^{00}_{\underline{\bar{a}}c\bar{b}}) + a\,.\,(\ldots) + b\,.\,(\ldots)\\
\sqsubseteq_Q\ & \bar{a}\,.\,(C^1_{\underline{c}ad}\|A^{11}_{\underline{b}d\bar{a}}\|c\,.\,C^1_{\underline{b}\bar{a}c}\|A^{00}_{\underline{\bar{a}}c\bar{b}})
\end{aligned}
$$

Let

$$F \stackrel{\text{def}}{=} \bar{a}\,.\,(C^1_{\underline{c}ad}\|A^{11}_{\underline{b}d\bar{a}}\|c\,.\,C^1_{\underline{b}\bar{a}c}\|A^{00}_{\underline{\bar{a}}c\bar{b}})$$

By substituting the results above in the last expression for M_2 we have

$$
\begin{aligned}
M_2 \sqsubseteq_Q\ & \tau_H(d\,.\,\bar{b}\,.\,(C^1_{\underline{c}ad}\|A^{01}_{\underline{b}d\bar{a}}\|C^0_{\underline{b}\bar{a}c}\|A^{10}_{\underline{\bar{a}}c\bar{b}})+\\
& \bar{b}\,.\,(d\,.\,(C^1_{\underline{c}ad}\|A^{01}_{\underline{b}d\bar{a}}\|C^0_{\underline{b}\bar{a}c}\|A^{10}_{\underline{\bar{a}}c\bar{b}}) + b\,.\,(d\,.\,C^1_{\underline{c}ad}\|A^{10}_{\underline{b}d\bar{a}}\|\bar{a}\,.\,c\,.\,C^1_{\underline{b}\bar{a}c}\|A^{10}_{\underline{\bar{a}}c\bar{b}})))\\
\sqsubseteq_Q\ & \tau_H(d\,.\,\bar{b}\,.\,b\,.\,F + \bar{b}\,.\,(d\,.\,b\,.\,F + b\,.\,d\,.\,F))
\end{aligned}
$$

$$\begin{aligned} &\equiv_Q \tau_H(d \,.\, \bar{b} \,.\, b \,.\, F + \bar{b} \,.\, b \,.\, F) \\ &\equiv_Q \tau_H(\bar{b} \,.\, b \,.\, F) \end{aligned}$$

The third step is obtained using axiom $\mathbf{I}_{16}$ together with axioms $\mathbf{I}_{8,9}$ for the substitutions. This step would not have been possible without accepting the input action b mentioned above. The fourth step is the application of axiom $\mathbf{I}_2$. This gives two results:

$$\begin{aligned} M_1 &\sqsubseteq_Q b \,.\, \tau_H(c \,.\, (d \,.\, C^1_{\underline{c}ad} \| A^{00}_{\underline{b}d\bar{a}} \| C^0_{\underline{b}\bar{a}c} \| \bar{b} \,.\, A^{10}_{\underline{\bar{a}}c\bar{b}})) \\ &\sqsubseteq_Q b \,.\, \tau_H(c \,.\, \bar{b} \,.\, b \,.\, F) \\ &\equiv_Q b \,.\, \bar{b} \,.\, b \,.\, M^* \end{aligned}$$

$$\begin{aligned} M_2 &\sqsubseteq_Q \tau_H(\bar{b} \,.\, b \,.\, F) \\ &\equiv_Q \bar{b} \,.\, b \,.\, M^* \end{aligned}$$

where the only interesting case is the second step for M_1 in which we used axiom $\mathbf{I}_8$. The process M^* comes from the fact that $\tau_H(F) \equiv M^*$. The result of the argument above is

$$M_2 \sqsubseteq_Q \bar{b} \,.\, b \,.\, M^*$$

$$M_1 \sqsubseteq_Q b \,.\, \bar{b} \,.\, b \,.\, M^*$$

In particular, by substituting in the last expression for M_1^*,

$$M_1^* \sqsubseteq_Q b \,.\, M_2^* + a \,.\, \bar{b} \,.\, b \,.\, \bar{b} \,.\, b \,.\, M^*$$

We can now analyze M_2^*. The treatment is standard and the substitution of M_1 derives from syntactical equivalence.

$$\begin{aligned} M_2^* &\equiv_Q \tau_H(a \,.\, (c \,.\, d \,.\, C^1_{\underline{c}ad} \| A^{10}_{\underline{b}d\bar{a}} \| b \,.\, c \,.\, C^0_{\underline{b}\bar{a}c} \| A^{11}_{\underline{\bar{a}}c\bar{b}}) + b \,.\, (C^0_{\underline{c}ad} \| A^{00}_{\underline{b}d\bar{a}} \| c \,.\, C^0_{\underline{b}\bar{a}c} \,.\, A^{11}_{\underline{\bar{a}}c\bar{b}})) \\ &\sqsubseteq_Q a \,.\, \tau_H(c \,.\, d \,.\, C^1_{\underline{c}ad} \| A^{10}_{\underline{b}d\bar{a}} \| b \,.\, c \,.\, C^0_{\underline{b}\bar{a}c} \| A^{11}_{\underline{\bar{a}}c\bar{b}}) + b \,.\, \tau_H(C^0_{\underline{c}ad} \| A^{00}_{\underline{b}d\bar{a}} \| c \,.\, C^0_{\underline{b}\bar{a}c} \| A^{11}_{\underline{\bar{a}}c\bar{b}}) \\ &\equiv_Q a \,.\, M_1 + b \,.\, \tau_H(C^0_{\underline{c}ad} \| A^{00}_{\underline{b}d\bar{a}} \| c \,.\, C^0_{\underline{b}\bar{a}c} \| A^{11}_{\underline{\bar{a}}c\bar{b}}) \end{aligned}$$

Let

$$M_3^* \stackrel{\text{def}}{=} \tau_H(C^0_{\underline{c}ad} \| A^{00}_{\underline{b}d\bar{a}} \| c \,.\, C^0_{\underline{b}\bar{a}c} \| A^{11}_{\underline{\bar{a}}c\bar{b}})$$

We have just shown that

$$M_2^* \sqsubseteq_Q b \,.\, M_3^* + a \,.\, b \,.\, \bar{b} \,.\, b \,.\, M^*$$

$$\begin{aligned}
M_3^* \equiv_Q\ & \tau_H(c \,.\, (a \,.\, d \,.\, C^1_{\underline{c}ad} \| A^{00}_{\underline{b}d\bar{a}} \| C^0_{\underline{b}\bar{a}c} \| \bar{b} \,.\, A^{10}_{\underline{\bar{a}}c\bar{b}}) + a \,.\, (c \,.\, d \,.\, C^1_{\underline{c}ad} \| A^{00}_{\underline{b}d\bar{a}} \| c \,.\, C^0_{\underline{b}\bar{a}c} \| A^{11}_{\underline{\bar{a}}c\bar{b}}) + \\
& b \,.\, (\ldots)) \\
\sqsubseteq_Q\ & \tau_H(c \,.\, (a \,.\, d \,.\, C^1_{\underline{c}ad} \| A^{00}_{\underline{b}d\bar{a}} \| C^0_{\underline{b}\bar{a}c} \| \bar{b} \,.\, A^{10}_{\underline{\bar{a}}c\bar{b}}) + a \,.\, (c \,.\, d \,.\, C^1_{\underline{c}ad} \| A^{00}_{\underline{b}d\bar{a}} \| c \,.\, C^0_{\underline{b}\bar{a}c} \| A^{11}_{\underline{\bar{a}}c\bar{b}})) \\
\equiv_Q\ & \tau_H(c \,.\, (a \,.\, (d \,.\, C^1_{\underline{c}ad} \| A^{00}_{\underline{b}d\bar{a}} \| C^0_{\underline{b}\bar{a}c} \| \bar{b} \,.\, A^{10}_{\underline{\bar{a}}c\bar{b}}) + \bar{b} \,.\, (a \,.\, d \,.\, C^1_{\underline{c}ad} \| A^{00}_{\underline{b}d\bar{a}} \| C^0_{\underline{b}\bar{a}c} \| A^{10}_{\underline{\bar{a}}c\bar{b}}) + \\
& b \,.\, (\cdots)) + \\
& a \,.\, (c \,.\, (d \,.\, C^1_{\underline{c}ad} \| A^{00}_{\underline{b}d\bar{a}} \| C^0_{\underline{b}\bar{a}c} \| \bar{b} \,.\, A^{10}_{\underline{\bar{a}}c\bar{b}}) + a \,.\, (\ldots) + b \,.\, (\ldots))) \\
\sqsubseteq_Q\ & \tau_H(c \,.\, (a \,.\, (d \,.\, C^1_{\underline{c}ad} \| A^{00}_{\underline{b}d\bar{a}} \| C^0_{\underline{b}\bar{a}c} \| \bar{b} \,.\, A^{10}_{\underline{\bar{a}}c\bar{b}}) + \bar{b} \,.\, (a \,.\, d \,.\, C^1_{\underline{c}ad} \| A^{00}_{\underline{b}d\bar{a}} \| C^0_{\underline{b}\bar{a}c} \| A^{10}_{\underline{\bar{a}}c\bar{b}})) + \\
& a \,.\, (c \,.\, (d \,.\, C^1_{\underline{c}ad} \| A^{00}_{\underline{b}d\bar{a}} \| C^0_{\underline{b}\bar{a}c} \| \bar{b} \,.\, A^{10}_{\underline{\bar{a}}c\bar{b}}))) \\
\sqsubseteq_Q\ & \tau_H(c \,.\, (a \,.\, \bar{b} \,.\, b \,.\, F + \bar{b} \,.\, (a \,.\, d \,.\, C^1_{\underline{c}ad} \| A^{00}_{\underline{b}d\bar{a}} \| C^0_{\underline{b}\bar{a}c} \| A^{10}_{\underline{\bar{a}}c\bar{b}})) + \\
& a \,.\, c \,.\, \bar{b} \,.\, b \,.\, F)
\end{aligned}$$

The steps above are standard. The problem is that we have to eliminate internal actions. In the steps below we first eliminate the internal action from the rightmost term $a \,.\, c \,.\, \bar{b} \,.\, b \,.\, F$ by means of axioms $\mathbf{I}_{8,9,11}$, then we apply axiom $\mathbf{I}_{17}$ obtaining the third expression. The rest is simple application of axioms $\mathbf{I}_{3,4}$.

$$\begin{aligned}
M_3^* \sqsubseteq_Q\ & \tau_H(c \,.\, (a \,.\, \bar{b} \,.\, b \,.\, F + \bar{b} \,.\, (a \,.\, d \,.\, C^1_{\underline{c}ad} \| A^{00}_{\underline{b}d\bar{a}} \| C^0_{\underline{b}\bar{a}c} \| A^{10}_{\underline{\bar{a}}c\bar{b}})) + \\
& a \,.\, c \,.\, \bar{b} \,.\, b \,.\, F) \\
\equiv_Q\ & \tau_H(c \,.\, (a \,.\, \bar{b} \,.\, b \,.\, F + \bar{b} \,.\, (a \,.\, d \,.\, C^1_{\underline{c}ad} \| A^{00}_{\underline{b}d\bar{a}} \| C^0_{\underline{b}\bar{a}c} \| A^{10}_{\underline{\bar{a}}c\bar{b}})) + \\
& a \,.\, \bar{b} \,.\, b \,.\, F) \\
\equiv_Q\ & \tau_H(a \,.\, \bar{b} \,.\, b \,.\, F + \bar{b} \,.\, (a \,.\, d \,.\, C^1_{\underline{c}ad} \| A^{00}_{\underline{b}d\bar{a}} \| C^0_{\underline{b}\bar{a}c} \| A^{10}_{\underline{\bar{a}}c\bar{b}})) \\
\equiv_Q\ & \tau_H(a \,.\, \bar{b} \,.\, b \,.\, F) + \tau_H(\bar{b} \,.\, (a \,.\, d \,.\, C^1_{\underline{c}ad} \| A^{00}_{\underline{b}d\bar{a}} \| C^0_{\underline{b}\bar{a}c} \| A^{10}_{\underline{\bar{a}}c\bar{b}})) \\
\equiv_Q\ & a \,.\, \bar{b} \,.\, b \,.\, M^* + \bar{b} \,.\, \tau_H(a \,.\, d \,.\, C^1_{\underline{c}ad} \| A^{00}_{\underline{b}d\bar{a}} \| C^0_{\underline{b}\bar{a}c} \| A^{10}_{\underline{\bar{a}}c\bar{b}})
\end{aligned}$$

Let

$$M_4^* \stackrel{\text{def}}{=} \tau_H(a \,.\, d \,.\, C^1_{\underline{c}ad} \| A^{00}_{\underline{b}d\bar{a}} \| C^0_{\underline{b}\bar{a}c} \| A^{10}_{\underline{\bar{a}}c\bar{b}})$$

We have just shown that

$$M_3^* \sqsubseteq_Q \bar{b} \,.\, M_4^* + a \,.\, \bar{b} \,.\, b \,.\, M^*$$

$$
\begin{aligned}
M_4^* &\equiv_Q \tau_H(a\,.\,(d\,.\,C^1_{\underline{c}ad}\|A^{00}_{\underline{b}d\bar{a}}\|C^0_{\underline{b}\bar{a}c}\|A^{10}_{\underline{\bar{a}}c\bar{b}}) + b\,.\,(a\,.\,d\,.\,C^1_{\underline{c}ad}\|A^{10}_{\underline{b}d\bar{a}}\|\bar{a}\,.\,c\,.\,C^1_{\underline{b}\bar{a}c}\|A^{10}_{\underline{\bar{a}}c\bar{b}}))\\
&\equiv_Q \tau_H(a\,.\,(d\,.\,(C^1_{\underline{c}ad}\|A^{01}_{\underline{b}d\bar{a}}\|C^0_{\underline{b}\bar{a}c}\|A^{10}_{\underline{\bar{a}}c\bar{b}}) + a\,.\,(\ldots)+\\
&\quad b\,.\,(d\,.\,C^1_{\underline{c}ad}\|A^{10}_{\underline{b}d\bar{a}}\|\bar{a}\,.\,c\,.\,C^1_{\underline{b}\bar{a}c}\|A^{10}_{\underline{\bar{a}}c\bar{b}}))) + b\,.\,M\\
&\sqsubseteq_Q \tau_H(a\,.\,(d\,.\,(C^1_{\underline{c}ad}\|A^{01}_{\underline{b}d\bar{a}}\|C^0_{\underline{b}\bar{a}c}\|A^{10}_{\underline{\bar{a}}c\bar{b}}) + b\,.\,(d\,.\,C^1_{\underline{c}ad}\|A^{10}_{\underline{b}d\bar{a}}\|\bar{a}\,.\,c\,.\,C^1_{\underline{b}\bar{a}c}\|A^{10}_{\underline{\bar{a}}c\bar{b}})) + b\,.\,M
\end{aligned}
$$

The steps above are standard. We now simplify the left expression.

$$
\begin{aligned}
&\tau_H(a\,.\,(d\,.\,(C^1_{\underline{c}ad}\|A^{01}_{\underline{b}d\bar{a}}\|C^0_{\underline{b}\bar{a}c}\|A^{10}_{\underline{\bar{a}}c\bar{b}}) + b\,.\,(d\,.\,C^1_{\underline{c}ad}\|A^{10}_{\underline{b}d\bar{a}}\|\bar{a}\,.\,c\,.\,C^1_{\underline{b}\bar{a}c}\|A^{10}_{\underline{\bar{a}}c\bar{b}}))\\
\equiv_Q\ &\tau_H(a\,.\,(d\,.\,(a\,.\,(\ldots) + b\,.\,(C^1_{\underline{c}ad}\|\bar{a}\,.\,A^{11}_{\underline{b}d\bar{a}}\|\bar{a}\,.\,c\,.\,C^1_{\underline{b}\bar{a}c}\|A^{10}_{\underline{\bar{a}}c\bar{b}}))+\\
&b\,.\,(d\,.\,(C^1_{\underline{c}ad}\|\bar{a}\,.\,A^{11}_{\underline{b}d\bar{a}}\|\bar{a}\,.\,c\,.\,C^1_{\underline{b}\bar{a}c}\|A^{10}_{\underline{\bar{a}}c\bar{b}}) + a\,.\,(\ldots) + b\,.\,(\ldots))))\\
\sqsubseteq_Q\ &\tau_H(a\,.\,(d\,.\,b\,.\,(C^1_{\underline{c}ad}\|\bar{a}\,.\,A^{11}_{\underline{b}d\bar{a}}\|\bar{a}\,.\,c\,.\,C^1_{\underline{b}\bar{a}c}\|A^{10}_{\underline{\bar{a}}c\bar{b}})+\\
&b\,.\,d\,.\,(C^1_{\underline{c}ad}\|\bar{a}\,.\,A^{11}_{\underline{b}d\bar{a}}\|\bar{a}\,.\,c\,.\,C^1_{\underline{b}\bar{a}c}\|A^{10}_{\underline{\bar{a}}c\bar{b}})))\\
\sqsubseteq_Q\ &\tau_H(a\,.\,(d\,.\,b\,.\,\bar{a}\,.\,(C^1_{\underline{c}ad}\|A^{11}_{\underline{b}d\bar{a}}\|c\,.\,C^1_{\underline{b}\bar{a}c}\|A^{00}_{\underline{\bar{a}}c\bar{b}})+\\
&b\,.\,d\,.\,\bar{a}\,.\,(C^1_{\underline{c}ad}\|A^{11}_{\underline{b}d\bar{a}}\|c\,.\,C^1_{\underline{b}\bar{a}c}\|A^{00}_{\underline{\bar{a}}c\bar{b}})))\\
\equiv_Q\ &\tau_H(a\,.\,b\,.\,\bar{a}\,.\,(C^1_{\underline{c}ad}\|A^{11}_{\underline{b}d\bar{a}}\|c\,.\,C^1_{\underline{b}\bar{a}c}\|A^{00}_{\underline{\bar{a}}c\bar{b}}))\\
\equiv_Q\ &a\,.\,b\,.\,M^*
\end{aligned}
$$

The fourth step above is justified from the fact that $C^1_{\underline{c}ad}\|\bar{a}\,.\,A^{11}_{\underline{b}d\bar{a}}\|\bar{a}\,.\,c\,.\,C^1_{\underline{b}\bar{a}c}\|A^{10}_{\underline{\bar{a}}c\bar{b}}$ is M' and the inequality derived at the beginning of this proof. The successive step is the application of axiom $\mathbf{I}_{16}$.

By substituting in the last expression obtained for M_4^* we have

$$M_4^* \sqsubseteq_Q b\,.\,M + a\,.\,b\,.\,M^*$$

We can now apply the recursive substitutivity axiom obtaining our conclusion, i.e., $M \sqsubseteq_Q S$.

■

Chapter 7

Conclusion

We have presented a process algebra (DIOA) with the following features: explicit interfaces associated with each expression, clear distinction between locally and globally controlled actions, input enabling, and actions under the control of at most one process. DIOA is directly related to I/O automata of Lynch and Tuttle [LT87], which have been successfully used for the verification of algorithms in distributed environments.

We have found a set of sound laws for the quiescent preorder over DIOA that are complete for recursion-free processes.

We have investigated the possibilities of using the quiescent preorder as an implementation relation and we have provided an intuitive understanding of its use. As a side effect we have found an intuitive property that could be required of a system and is not detected by the quiescent and fair preorders.

We have given two simple example specifications to show how axioms can be used to prove correctness of implementations. The use of axioms, as can be seen in the given examples, seems sometimes simpler than the method based on possibilities mappings, that is characteristic for I/O automata, in the sense that the specification itself helps the verifier in understanding the axioms that need to be applied.

The above results, however, make clear that there are still many open problems. Some of the problems are understanding when algebraic reasoning is really simpler than the method based on possibilities mappings, whether it is possible to use algebraic reasoning on very complex systems, whether it is possible to integrate algebraic reasoning with simulation techniques in order to simplify correctness proofs. For the last topic a useful fact is that most of the presented axioms are still valid if the underlying model deals with infinite traces or with fair traces.

An advantage of the algebraic method we have presented is that it seems easy to be mechanized. A proposal of research could involve an understanding of how such a mechanized system could work. The tools could deal both with algebraic and mapping based methods and could be a sort of interactive environment where the user is helped in providing correctness proofs or discovering errors.

A third open problem is finding general formalisms capturing the essence of I/O automata without being necessarily input enabled. Input enabling, in fact, is one of the most discussed features of the I/O automaton model since many reasonable concurrent tasks cannot be described at a sufficient abstract level using I/O automata. In this thesis we have investigated the implications of input enabling on the algebraic laws of a generic process algebra; the successive step is verifying how the notion of input enabling could be embedded into a generic process algebra without the input enabling condition. In doing so, we obtain a more expressive model having all the features of I/O automata when a process meets the input enabling condition. Moreover, we can understand the essence of the commonly used implementation relations by viewing them through the process algebraic framework and by comparing them with the relations that are commonly used within process algebras. Some relations that seem very closed to the preorder relations of I/O automata and that deserve further investigation are the testing preorders of De Nicola and Hennessy [DH84, De 85a, Hen88].

Although the above topics are quite important, we believe that one of the most important topics is to give a strong foundation to the commonly used verification methods. For example, in Chapter 6 we have given an informal description of how and when the quiescent preorder could be thought as an implementation relation; in [LT87] Nancy Lynch and Mark Tuttle give an informal understanding of how the fair preorder can be used as an implementation relation; in Chapter 6 we have given an example of a property that could be required of a system and is not detected by the fair preorder. The questions are then straightforward: What do we require to an implementation relation? What are the properties we are interested in? What properties does a particular relation guarantee to be preserved? What is a property? Trying to give an answer to the questions above is definitely worth doing and should be one of the main topics for a long term plan of further research.

Appendix A

Tables

Name	Op.	Domain	Range	Restrictions
quiescent	nil_S	λ	S	
omega	Ω_S	λ	S	
prefixing	$a._S$	S	S	$a \in ext(S)$
ichoice	$\oplus_S$	S, S	S	
echoice	${}_I+_J^S$	S, S	S	$I, J \subseteq in(S)$
parallel	${}_{S_1}\|_{S_2}$	S_1, S_2	S_3	$out(S_1) \cap out(S_2) = \emptyset$ $out(S_3) = out(S_1) \cup out(S_2)$ $in(S_3) = (in(S_1) \cup in(S_2)) \setminus out(S_3)$
hiding	τ_I^S	S	S'	$I \subseteq out(S), S' = (in(S), out(S) \setminus I)$
renaming	ρ_S	S	S'	for each injective $\rho : acts(S) \longrightarrow acts(S')$ $S' = (\rho(in(S)), \rho(out(S)))$
process	X_S	λ	S	$X_S \in \mathcal{X}_S$

Table A.1: The signature of DIOA

nil	$nil_S \xrightarrow{a} \Omega_S$	$\forall a \in in(S)$			
ome$_1$	$\Omega_S \xrightarrow{a} \Omega_S$	$a \in ext(S)$	**ome**$_2$	$\Omega_S \xrightarrow{\tau} nil_S$	
pre$_1$	$a \,._S\, e \xrightarrow{a} e$		**pre**$_2$	$a \,._S\, e \xrightarrow{b} \Omega_S$	$\forall b \in in(S)\backslash\{a\}$
ich$_1$	$e_1 \oplus_S e_2 \xrightarrow{\tau} e_1$		**ich**$_2$	$e_1 \oplus_S e_2 \xrightarrow{\tau} e_2$	
ich$_3$	$\dfrac{e_1 \xrightarrow{a} e_1'}{e_1 \oplus_S e_2 \xrightarrow{a} e_1'}$	$\forall a \in in(S)$	**ich**$_4$	$\dfrac{e_2 \xrightarrow{a} e_2'}{e_1 \oplus_S e_2 \xrightarrow{a} e_2'}$	$\forall a \in in(S)$
ech$_1$	$\dfrac{e_1 \xrightarrow{a} e_1'}{e_1 \,{}_I{+}_J^S\, e_2 \xrightarrow{a} e_1'}$	$\forall a \in I \cup out(S)$			
ech$_2$	$\dfrac{e_2 \xrightarrow{a} e_2'}{e_1 \,{}_I{+}_J^S\, e_2 \xrightarrow{a} e_2'}$	$\forall a \in J \cup out(S)$			
ech$_3$	$e_1 \,{}_I{+}_J^S\, e_2 \xrightarrow{a} \Omega_S$	$\forall a \in in(S)\backslash(I \cup J)$			
ech$_4$	$\dfrac{e_1 \xrightarrow{\tau} e_1'}{e_1 \,{}_I{+}_J^S\, e_2 \xrightarrow{\tau} e_1' \,{}_I{+}_J^S\, e_2}$				
ech$_5$	$\dfrac{e_2 \xrightarrow{\tau} e_2'}{e_1 \,{}_I{+}_J^S\, e_2 \xrightarrow{\tau} e_1' \,{}_I{+}_J^S\, e_2'}$				
tau$_1$	$\dfrac{e \xrightarrow{a} e'}{\tau_I^S(e) \xrightarrow{a} \tau_I^S(e')}$	$a \notin I$	**tau**$_2$	$\dfrac{e \xrightarrow{a} e'}{\tau_I^S(e) \xrightarrow{\tau} \tau_I^S(e')}$	$a \in I$
rho	$\dfrac{e \xrightarrow{a} e'}{\rho_S(e) \xrightarrow{\rho(a)} \rho_S(e')}$				
par$_1$	$\dfrac{e_1 \xrightarrow{a} e_1' \qquad e_2 \xrightarrow{a} e_2'}{e_1 \,{}_{S_1}\|_{S_2}\, e_2 \xrightarrow{a} e_1' \,{}_{S_1}\|_{S_2}\, e_2'}$				
par$_2$	$\dfrac{e_1 \xrightarrow{a} e_1'}{e_1 \,{}_{S_1}\|_{S_2}\, e_2 \xrightarrow{a} e_1' \,{}_{S_1}\|_{S_2}\, e_2}$	$a \in acts(S_1)\backslash ext(S_2)$			
par$_3$	$\dfrac{e_2 \xrightarrow{a} e_2'}{e_1 \,{}_{S_1}\|_{S_2}\, e_2 \xrightarrow{a} e_1 \,{}_{S_1}\|_{S_2}\, e_2'}$	$a \in acts(S_2)\backslash ext(S_1)$			
rec	$\dfrac{e \xrightarrow{a} e'}{X \xrightarrow{a} e'}$	if $X \stackrel{\text{def}}{=} e$			

Table A.2: The transition rules for DIOA

$$wsi_{A,B}(nil) = \emptyset$$

$$wsi_{A,B}(\Omega) = \emptyset$$

$$wsi_{A,B}(a\,.\,e) = \begin{cases} \{a\} & \text{if } a \in in(e)\backslash A \\ \emptyset & \text{if } a \in out(e) \cup A \end{cases}$$

$$wsi_{A,B}(e_1 \oplus e_2) = wsi_{A,B}(e_1) \cap wsi_{A,B}(e_2)$$

$$wsi_{A,B}(e_1 \;{}_I{+}_J\; e_2) = \begin{cases} \emptyset & \text{if } B \cap A \cap (in(e_1)\backslash(I \cup J)) \neq \emptyset \\ (I \cap wsi_{A,B\cap(I\cup out(e_1))}(e_1)) \cup (J \cap wsi_{A,B\cap(J\cup out(e_2))}(e_2)) & \\ \quad \text{otherwise} & \end{cases}$$

$$wsi_{A,B}(\tau_I(e)) = wsi_{A\cup I,B}(e)$$

$$wsi_{A,B}(\rho(e)) = \rho(wsi_{\rho^{-1}(A),\rho^{-1}(B)}(e))$$

$$wsi_{A,B}(e_1 \| e_2) = wsi_{\emptyset,\emptyset}(e_1) \cup wsi_{\emptyset,\emptyset}(e_2)$$

$$wsi_{A,B}(X) = wsi_{A,B}(E(X))$$

Table A.3: Definition of *wsi* for DIOA. $wsi(e) \stackrel{\text{def}}{=} wsi_{\emptyset,\emptyset}(e)$

$$wso_{A,B}(nil) = \begin{cases} \emptyset & \text{if } A \cap B \cap in(nil) = \emptyset \\ out(nil) \backslash A & \text{otherwise} \end{cases}$$

$$wso(\Omega) = out(\Omega) \backslash A$$

$$wso_{A,B}(a\,.\,e) = \begin{cases} out(e) \backslash A & \text{if } B \cap A \cap \overline{\{a\}} \neq \emptyset \\ \{a\} \cap out(e) & \text{if } B \cap A \cap \overline{\{a\}} = \emptyset \text{ and } a \notin A \\ wso_{A,A}(e) & \text{if } B \cap A \cap \overline{\{a\}} = \emptyset \text{ and } a \in A \cap B \\ \emptyset & \text{if } B \cap A \cap \overline{\{a\}} = \emptyset \text{ and } a \in A \backslash B \end{cases}$$

$$wso_{A,B}(e_1 \oplus e_2) = wso_{A,B}(e_1) \cup wso_{A,B}(e_2)$$

$$wso_{A,B}(e_1 \; {}_I{+}_J \; e_2) = \begin{cases} wso_{A,B\cap(I\cup out(e_1))}(e_1) \cup wso_{A,B\cap(J\cup out(e_2))}(e_2) & \text{if } B \cap A \cap \overline{I \cup J} = \emptyset \\ out(e_1) \backslash A & \text{otherwise} \end{cases}$$

$$wso_{A,B}(\tau_I(e)) = wso_{A\cup I,B\cup I}(e)$$

$$wso_{A,B}(\rho(e)) = \rho(wso_{\rho^{-1}(A),\rho^{-1}(B)}(e))$$

$$wso_{A,B}(e_1 \| e_2) = \begin{cases} wso_{A,A}(e_1) \cup wso_{A,A}(e_2) & \text{if } \exists a \in B \cap A : a \in acts(e_1) \backslash ext(e_2) \\ & \quad \text{or } a \in acts(e_2) \backslash ext(e_1) \\ wso_{A,B}(e_1) \cup wso_{A,B}(e_2) & \text{otherwise} \end{cases}$$

$$wso_{A,B}(X) = wso_{A,B}(E(X))$$

Table A.4: Definition of *wso* for DIOA $wso(e) \stackrel{\text{def}}{=} wso_{\emptyset,\emptyset}(e)$

$localen(nil) = \emptyset$
$localen(a.e) = \{a\} \cap out(e)$
$localen(e_1 \oplus e_2) = localen(e_1) \cup localen(e_2) \cup \{\tau\}$
$localen(e_1 \; {}_I{+}_J \; e_2) = localen(e_1) \cup localen(e_2)$
$localen(\tau_I(e)) = localen(e)$
$localen(\rho(e)) = \rho(localen(e))$
$localen(e_1 \| e_2) = localen(e_1) \cup localen(e_2)$
$localen(X) = localen(E(X))$
$inten(e) = true$ iff $\{\tau\} \in localen(e)$
$quiet(e) = true$ iff $localen(e) = \emptyset$

Table A.5: Definition of *localen*, *inten* and *quiet*

	renaming axioms
$\mathbf{R}_1$	$\rho(nil) \equiv_Q nil$
$\mathbf{R}_2$	$\rho(a \,.\, e) \equiv_Q \rho(a) \,.\, \rho(e)$
$\mathbf{R}_3$	$\rho(e \oplus f) \equiv_Q \rho(e) \oplus \rho(f)$
$\mathbf{R}_3$	$\rho(e \;{}_I{+}_J\; f) \equiv_Q \rho(e) \;{}_{\rho(I)}{+}_{\rho(J)}\; \rho(f)$
$\mathbf{R}_4$	$\rho_1(\rho_2(e)) \equiv_Q \rho_1 \circ \rho_2(e)$
$\mathbf{R}_5$	$\rho(\tau_I(e)) \equiv_Q \tau_{\rho'(I)}(\rho'(e))$ if ρ' extends ρ
$\mathbf{R}_6$	$\rho(e \Vert f) \equiv_Q \rho(e) \Vert \rho(f)$
	parallel axioms
$\mathbf{P}_1$	$e \Vert f \equiv_Q f \Vert e$
$\mathbf{P}_2$	$(e \Vert f) \Vert g \equiv_Q e \Vert (f \Vert g)$
$\mathbf{P}_3$	$\Omega_{S_1} \Vert nil_{S_2} \sqsubseteq_Q \Omega_{S_3} \Vert nil_{S_4}$ if $(out(S_1) \subseteq out(S_3)) \wedge ((in(S_2) \subseteq in(S_4)) \vee out(S_4) = \emptyset)$
	external choice axioms
$\mathbf{Ec}_1$	$e \;{}_I{+}_J\; f \equiv_Q f \;{}_J{+}_I\; e$
$\mathbf{Ec}_2$	$(e \;{}_I{+}_J\; f) \;{}_{I \cup J}{+}_K\; g \equiv_Q e \;{}_I{+}_{J \cup K}\; (f \;{}_J{+}_K\; g)$
$\mathbf{Ec}_3$	$e \equiv_Q e \;{}_I{+}_J\; e$ if $Wsi(e) \subseteq I \cup J$
$\mathbf{Ec}_4$	$e \;{}_I{+}_J\; f \equiv_Q (e \;{}_H{+}_K\; e) \;{}_I{+}_J\; f$ if $I \subseteq H \cup K$
$\mathbf{Ec}_5$	$\dfrac{(not(quiet(e)) \wedge not(inten(e))) \vee quiet(f)}{e \sqsubseteq_Q e \;{}_I{+}_J\; f}$ if $J \cap Wsi(f) \subseteq I$
$\mathbf{Ec}_6$	$\dfrac{(not(quiet(e)) \wedge not(inten(e))) \vee quiet(f)}{e \;{}_I{+}_J\; g \sqsubseteq_Q (e \;{}_H{+}_K\; f) \;{}_I{+}_J\; g}$ if $K \cap Wsi(f) \cap I \subseteq H$
$\mathbf{Ec}_7$	$\dfrac{quiet(f)}{e \;{}_I{+}_J\; f \sqsubseteq_Q e}$ if $Wsi(e) \subseteq I$ and $Wsi(e) \cap J = \emptyset$
$\mathbf{Ec}_8$	$\dfrac{quiet(f)}{(e \;{}_H{+}_K\; f) \;{}_I{+}_J\; g \sqsubseteq_Q e \;{}_I{+}_J\; g}$ if $Wsi(e) \cap I \subseteq H$ and $K \cap Wsi(e) \cap I = \emptyset$
$\mathbf{Ec}_9$	$e \equiv_Q e \;{}_I{+}_J\; a \,.\, \Omega$ if $Wsi(e) \subseteq I$ and $Wsi(e) \cap J = \emptyset$
$\mathbf{Ec}_{10}$	$a \,.\, e \;{}_I{+}_J\; a \,.\, f \equiv_Q a \,.\, (e \oplus f)$ if $a \in out(e) \cup (I \cap J)$
$\mathbf{Ec}_{11}$	$e \;{}_I{+}_J\; f \sqsubseteq_Q e \oplus f$ where $Wsi(e) \cap Wsi(f) \subseteq I \cup J$

Table A.6: The axioms for DIOA.

Ec$_{12}$ $\dfrac{quiet(e) \iff quiet(f) \wedge not(inten(e)) \wedge not(inten(f))}{e\ {}_I{+}_J\ f \equiv_Q e \oplus f}$ if $Wsi(e) \cup Wsi(f) \subseteq I \cap J$

Ec$_{13}$ $\dfrac{a \in in(e) \vee (not(quiet(q)) \wedge not(inten(q))) \vee quiet(f)}{(a\,.\,e\ {}_I{+}_J\ f) \oplus g \equiv_Q (a\,.\,e\ {}_I{+}_J\ f) \oplus (a\,.\,e\ {}_I{+}_K\ g)}$ if $Wsi(g) \subseteq K$, and $\{a\} \cap I \subseteq \{a\} \cap K$

Ec$_{14}$ $e\ {}_I{+}_J\ f \equiv_Q e\ {}_{I \setminus \{a\}}{+}_{J \setminus \{a\}}\ f$ if $a \in I \setminus Wsi(e)$.

Ec$_{15}$ $\dfrac{quiet(f)}{e \equiv_Q e\ {}_I{+}_\emptyset\ f}$ where $Wsi(e) \subseteq I$

Ec$_{16}$ $\dfrac{quiet(f)}{e\ {}_I{+}_J\ g \equiv_Q (e\ {}_I{+}_K\ f)\ {}_I{+}_J\ g}$ if $K \cap I = \emptyset$

internal choice axioms

Ic$_1$ $e \oplus f \equiv_Q f \oplus e$

Ic$_2$ $(e \oplus f) \oplus g \equiv_Q e \oplus (f \oplus g)$

Ic$_3$ $e \equiv_Q e \oplus e$

Ic$_4$ $a\,.\,(e \oplus f) \equiv_Q a\,.\,e \oplus a\,.\,f$

Ic$_5$ $(e \oplus f)\ {}_I{+}_J\ g \equiv_Q (e\ {}_I{+}_J\ g) \oplus (f\ {}_I{+}_J\ g)$

Ic$_6$ $\tau_I(e \oplus f) \equiv_Q \tau_I(e) \oplus \tau_I(f)$

Ic$_7$ $(e \oplus f) \| g \equiv_Q (e \| g) \oplus (f \| g)$

Ic$_8$ $e \sqsubseteq_Q e \oplus f$

hiding axioms

I$_1$ $\tau_\emptyset(e) \equiv_Q e$

I$_2$ $\tau_I(nil) \equiv_Q nil$

I$_3$ $\tau_I(a\,.\,e) \equiv_Q a\,.\,\tau_I(e)$ if $a \notin I$

I$_4$ $\tau_I(e\ {}_H{+}_K\ f) \equiv_Q \tau_I(e)\ {}_H{+}_K\ \tau_I(f)$ if $Wso(e) \cap I = Wso(f) \cap I = \emptyset$

I$_5$ $\tau_I(\tau_J(e)) \equiv_Q \tau_{I \cup J}(e)$

I$_6$ $\tau_I(e) \| \tau_J(f) \equiv_Q \tau_{I \cup J}(e \| f)$ if $I \cap acts(f) = J \cap acts(e) = \emptyset$

I$_7$ $e \equiv_Q \rho(e)$ if ρ is the identity function

I$_8$ $\dfrac{\tau_I(e) \sqsubseteq_Q \tau_I(f)}{\tau_I(a\,.\,e) \sqsubseteq_Q \tau_I(a\,.\,f)}$

Table A.7: The axioms for DIOA: actions of the form τ_i belong to I.

$\mathbf{I}_9$ $\dfrac{\tau_I(e) \sqsubseteq_Q \tau_I(g)}{\tau_I(e\ {}_H{+}_K\ f) \sqsubseteq_Q \tau_I(g\ {}_H{+}_K\ f)}$

$\mathbf{I}_{10}$ $\tau_I(e) \sqsubseteq_Q \tau_I(i\,.\,e\ {}_H{+}_K\ f)$

$\mathbf{I}_{11}$ $\tau_I(i\,.\,e) \equiv_Q \tau_I(e)$ if $Wsi(e) = \emptyset$

$\mathbf{I}_{12}$ $\dfrac{not(quiet(e)) \quad not(inten(e))}{\tau_I(e\ {}_H{+}_\emptyset\ i\,.\,f) \equiv_Q \tau_I(e \oplus f)}$ if $Wsi(e) \subseteq H$

$\mathbf{I}_{13}$ $\dfrac{quiet(e)}{\tau_I(e\ {}_H{+}_\emptyset\ i\,.\,f) \equiv_Q \tau_I(e\ {}_K{+}_K\ f)}$ if $Wsi(e) \subseteq H$ and $Wsi(e) \subseteq K$

$\mathbf{I}_{14}$ $\tau_I((\Omega_{S_0} \| nil_{S_1} \| \cdots \| nil_{S_n}) \| e) \equiv_Q \tau_I(\Omega \| e)$ if $\forall_{1 \leq j \leq n} (out(S_0) \cap in(S_j) \cap I) \backslash in(e) \neq \emptyset$

$\mathbf{I}_{15}$ $\tau_I(\Omega_{S_0} \| nil_{S_1} \| \cdots \| nil_{S_n}) \equiv_Q \Omega_{S_0 \backslash I} \| nil_{S_1 \backslash I} \| \cdots \| nil_{S_n \backslash I}$ if $\forall_{1 \leq i \leq n} out(S_0) \cap in(S_i) \cap I = \emptyset$

$\mathbf{I}_{16}$ $\tau_I(a\,.\,i\,.\,e\ {}_{\{a\} \cap in(e)}{+}_\emptyset\ i\,.\,a\,.\,e) \equiv_Q \tau_I(a\,.\,e)$ if $Wsi(e) = \emptyset$

$\mathbf{I}_{17}$ $\tau_I(i\,.\,(e\ {}_\emptyset{+}_J\ f)\ {}_\emptyset{+}_J\ f) \equiv_Q \tau_I(e\ {}_\emptyset{+}_J\ f)$ if $quiet(f)$ and $Wsi(f) \subseteq J$

omega axioms

R $\rho(\Omega_S) \equiv_Q \Omega_{\rho(S)}$ **M** $e \sqsubseteq_Q \Omega$

I $\tau_I(\Omega_S) \equiv_Q \Omega_{S'}$ where $S' = (in(S), out(S) \backslash I)$

P $\Omega_{S_1} \| \Omega_{S_2} \equiv_Q \Omega_{S_3}$ where S_3 is the composition of S_1 and S_2

expansion axioms

$\mathbf{E}_1$ Let $e \equiv \Omega_{S_0} \| nil_{s_1} \| \cdots \| nil_{S_n}$ be of sort S. For each $a \in out(S_0) \cup in(S)$ let e_a be the state that e reaches with action a. Then $e \equiv_Q (\sum_{a \in out(S_0) \cup in(S)} a\,.\,e_a) \oplus (\sum_{a \in in(S)} a\,.\,e_a)$.

$\mathbf{E}_2$ Let $e \equiv e_1 \| e_2 \| \cdots \| e_n$ where each e_i is of the form $\sum_j a_{i\,j}\,.\,e_{i\,j}$. For each action $a \in ext(e)$ let

$$E_a^i = \begin{cases} \{e_{i\,j} | a_{i\,j} = a\} & \text{if } a \in acts(e_i) \\ \{e_i\} & \text{otherwise} \end{cases}$$

Let $out(a)$ be the index j s.t. a is an output action of j (0 otherwise) and let

$$E_a = \begin{cases} \emptyset & \text{if } out(a) \neq 0 \text{ and } E_a^{out(a)} = \emptyset \\ \{f_1 \| \cdots \| f_n : f_i \in E_a^i \vee (E_a^i = \emptyset \wedge f_i \equiv \Omega)\} & \text{otherwise} \end{cases}$$

Then $e \equiv_Q \sum_{a \in ext(e)} (\sum_{f \in E_a} a.f)$.

$\mathbf{Cp}_1$ Let $e_i, 0 \leq i \leq n$ be atomic expressions and, for each action a, let f_i^a be the state that e_i reaches with action a ($\bullet$ if no state exists). Then $e_0 \sqsubseteq_Q \sum_{1 \leq i \leq n} e_i$ iff, for each action a, either $f_i^a \equiv e_i, 0 \leq i \leq n$ or $f_0^a \equiv \bullet$ or $f_0^a \sqsubseteq_Q \sum_{f_i^a \neq \bullet} f_i^a$.

Table A.8: The axioms for DIOA.

Bibliography

[ABV92] L. Aceto, B. Bloom, and F. Vaandrager. Turning SOS rules into equations. In *Proceedings of the Seventh Annual Symposium on Logic in Computer Science*, 1992.

[BIM90] B. Bloom, S. Istrail, and A.R. Meyer. Bisimulation can't be traced. Technical Report 90-1150, Department of Computer Science, Cornell University, Ithaca, New York, August 1990.

[BV88] J.C.M. Baeten and F.W. Vaandrager. Specification and verification of a circuit in ACP. Technical report, Department of Mathematics and Computer Science, Amsterdam, October 1988.

[BW90] J.C.M. Baeten and W.P Weijland. *Process Algebra.* Cambridge Tracts in Theoretical Computer Science 18, Cambridge University Press, 1990.

[De 84] R. De Simone. Calculabilité et expressivité dans l'algebra de processus parallèles MEIJE, 1984. Thèse de 3^e cycle, Univ. Paris 7.

[De 85a] R. De Nicola. *Testing Equivalences and Fully Abstract Models for Communicating Processes.* PhD thesis, Department of Computer Science, University of Edinburgh, 1985.

[De 85b] R. De Simone. Higher-level synchronising devices in MEIJE-SCCS. *Theoretical Computer Science*, 37:245–267, 1985.

[De 87] R. De Nicola. Extensional equivalences for transition systems. *Acta Informatica*, 24:211–237, 1987.

[DH84] R. De Nicola and M. Hennessy. Testing equivalences for processes. *Theoretical Computer Science*, 34:83–133, 1984.

[DH87] R. De Nicola and M. Hennessy. CCS without τ's. *LNCS*, 249:136–152, 1987.

[Fra86] Nissim Francez. *Fairness.* Springer-Verlag, Berlin, 1986.

[Gla90] R.J. van Glabbeek. The linear time – branching time spectrum. In J.C.M. Baeten and J.W. Klop, editors, *Proceedings CONCUR 90,* Amsterdam, volume 458 of *Lecture Notes in Computer Science*, pages 278–297. Springer-Verlag, 1990.

[GV89] J.F. Groote and F. Vaandrager. Structured operational semantics and bisimulation as a congruence (extended abstract). In G. Ausiello, M. Dezani-Ciancaglini, and S. Ronchi Della Rocca, editors, *Proceedings* 16^{th} *ICALP,* Stresa, volume 372 of *Lecture Notes in Computer Science*, pages 423–438. Springer-Verlag, 1989. Full version to appear in *Information and Computation.*

[Hen88] M. Hennessy. *Algebraic Theory of Processes.* MIT Press, Cambridge, Massachusetts, 1988.

[Hoa85] C.A.R. Hoare. *Communicating Sequential Processes.* Prentice-Hall International, Englewood Cliffs, 1985.

[Jon85] B. Jonsson. A model and proof system for asynchronous networks. In *Proceedings of the* 4^{th} *Annual ACM Symposium on Principles of Distributed Computing,* Minaki, Ontario, Canada, pages 49–58, 1985.

[Jon87] B. Jonsson. *Compositional Verification of Distributed Systems.* PhD thesis, Department of Computer Systems, Uppsala University, 1987. DoCS 87/09.

[Jos92] M.B. Josephs. Receptive process theory. *Acta Informatica*, 29:17–31, 1992.

[Kal87] A. Kaldewaij. The translation of processes into circuits. In *Proceedings PARLE Conference, LNCS 258*, pages 195–212, 1987.

[LT87] N.A. Lynch and M.R. Tuttle. Hierarchical correctness proofs for distributed algorithms. In *Proceedings of the 6^{th} Annual ACM Symposium on Principles of Distributed Computing*, pages 137–151, Vancouver, Canada, August 1987. A full version is available as MIT Technical Report MIT/LCS/TR-387.

[Mil89] R. Milner. *Communication and Concurrency.* Prentice-Hall International, Englewood Cliffs, 1989.

[Par81] D.M.R. Park. Concurrency and automata on infinite sequences. In P. Deussen, editor, *5^{th} GI Conference*, volume 104 of *Lecture Notes in Computer Science*, pages 167–183. Springer-Verlag, 1981.

[Plo81] G.D. Plotkin. A structural approach to operational semantics. Technical Report DAIMI FN-19, Computer science Department, Aarhus University, 1981.

[Seg91] R. Segala. Algebre di processi come automi con input e output, 1991. Tesi di laurea, Università di Pisa, Italy.

[Sta84] E.W. Stark. *Foundations of a theory of specification for Distributed Systems.* PhD thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, August 1984. Available as Technical Report MIT/LCS/TR-342.

[Vaa91] F.W. Vaandrager. On the relationship between process algebra and Input/Output automata. In *Proceedings of the Sixth Annual Symposium on Logic in Computer Science*, 1991.